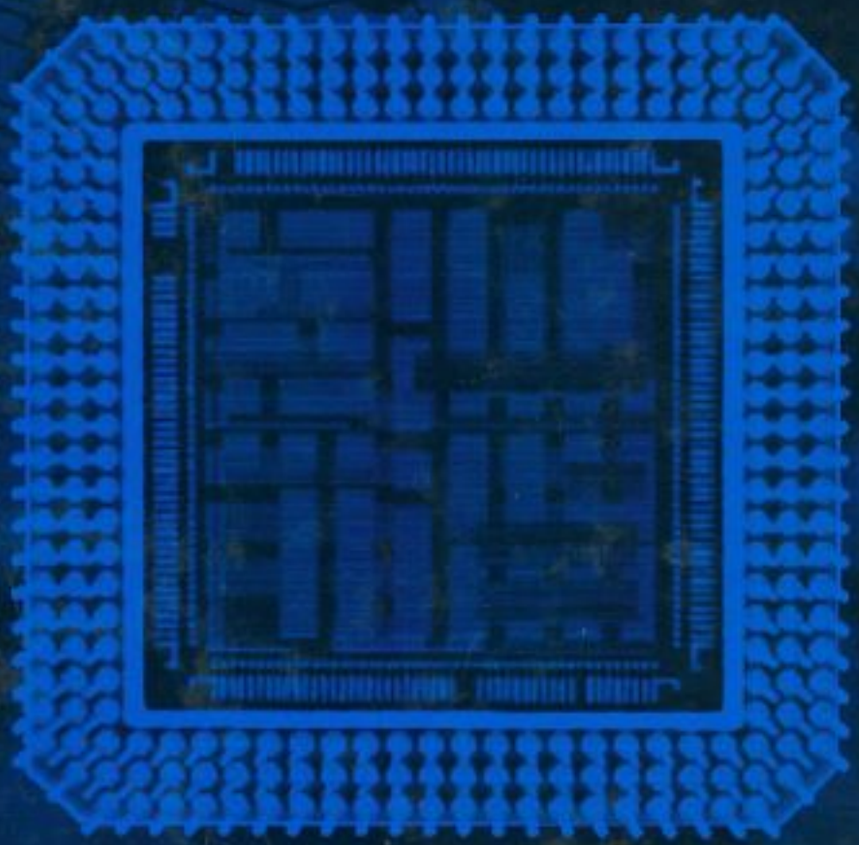
Copyrighted Material

Microelectronics

Second Edition



Jerry C. Whitaker

CRC Taylor & Francis
Taylor & Francis Group

Copyrighted Material

VISIT…

Microelectronics

Second Edition

© 2006 by Taylor & Francis Group, LLC

Microelectronics

Second Edition

Jerry C. Whitaker

Taylor & Francis
Taylor & Francis Group

Boca Raton London New York

A CRC title, part of the Taylor & Francis imprint, a member of the
Taylor & Francis Group, the academic division of T&F Informa plc.

© 2006 by Taylor & Francis Group, LLC

This material was previously published in *The Electronics Handbook, Second Edition*. © CRC Press LLC 2005.

Published in 2006 by
CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2006 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group

No claim to original U.S. Government works
Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-10: 0-8493-3391-1 (Hardcover)
International Standard Book Number-13: 978-0-8493-3391-0 (Hardcover)
Library of Congress Card Number 2005053102

This book contains information obtained from authentic and highly regarded sources. Reprinted material is quoted with permission, and sources are indicated. A wide variety of references are listed. Reasonable efforts have been made to publish reliable data and information, but the author and the publisher cannot assume responsibility for the validity of all materials or for the consequences of their use.

No part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (http://www.copyright.com/) or contact the Copyright Clearance Center, Inc. (CCC) 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Microelectronics / [edited by] Jerry C. Whitaker.-- 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-8493-3391-1 (alk. paper)
1. Microelectronics. I. Whitaker, Jerry C.

TK7874.M4587 2005
621.381--dc22 2005053102

informa
Taylor & Francis Group
is the Academic Division of Informa plc.

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

© 2006 by Taylor & Francis Group, LLC

Preface

The discipline of microelectronics has played a fundamental role in shaping the electronics industry, as well as related industries that rely on electronic components and subsystems. In a realm where changes happen frequently and dramatically, the constant themes that have persisted are miniaturization, increased speed, reduced power consumption, and reduced cost. These effects have resulted in an increased demand for microelectronics in all sectors of consumer, industrial, and military products. Advancements in manufacturing have enabled these devices to be produced in very high volumes, thereby reducing the cost per device. In turn, the lower cost fuels future demand which pushes the industry for further miniaturization and higher volume manufacturing.

The combination of reduced size, increased speed, and increased capacity of microelectronics devices was first observed by Gordon E. Moore (the legendary Chairman of Intel) who during the 1960s commented that the feature size of semiconductor transistors reduced by 10 percent per year. In fact, the reduction has been even more dramatic than that. The capacity of dynamic random access memory (DRAM) integrated circuits has quadrupled approximately every three years. The increased density of transistors contained in microelectronic devices has resulted in a phenomenon of virtually "free computing power."

The digital revolution of the 1980s ushered in the so-called *Information Age*, and with it came substantial growth of data recording systems, primarily associated with the desktop computer. The transition to digital systems is far from complete, but it has already had far-reaching impact. Perhaps most important is the nearly universal usability of digital information. Any form of expression that can be quantified can be turned into a digital bit stream, and carried in tandem with any other type of expression.

Computers manipulate data and in this context they can be thought of as the engines necessary to organize and access information. Computers are rapidly changing the world—from the workplace to the home—ranging from traditional stand-alone mainframes to embedded computational devices. Almost every piece of equipment or appliance contains one or more microprocessors.

The market demand for microelectronics has evolved from largely a military-driven demand to one that is now largely consumer-driven. Consequently, the device features have also been targeted at consumer needs, such as low power, low cost, and mass market applications, rather than military needs, such as meeting military specifications for reliability and packaging, specialized applications, and the resulting high cost of such devices. The performance of microelectronics is measured, thus, from the viewpoint of the technological aspects of the device, as well as from the viewpoint of end user effectiveness. The goal is to enable the end user of the devices to perform complex tasks in a more efficient manner than what was previously possible.

This Handbook focuses on the technological issues within specific microelectronic technologies and examines how they affect the push of technology that drives the next generation of microelectronics. The chapters describe the three primary elements of microelectronics technology: materials, devices, and applications.

This Handbook strives to give the reader a broad understanding of the technologies shaping microelectronics and how these technologies affect the end uses of the devices.

© 2006 by Taylor & Francis Group, LLC

Contributors

Samuel O. Agbo
California Polytechnic State University
San Luis Obispo, California

Constantine N. Anagnostopoulos
Microelectronics Technical Division
Eastman Kodak Company
Rochester, New York

Praveen Asthana
IBM Corporation
San Jose, California

David F. Besch
University of the Pacific
Stockton, California

Bruce W. Bomar
Department of Electrical and Computer Engineering
University of Tennessee Space Institute
Tullahoma, Tennessee

John R. Brews
University of Arizona
Tucson, Arizona

Paulo Cardieri
University of Campinas
São Paulo, Brazil

Jonathon A. Chambers
Cardiff School of Engineering
Cardiff University
Wales, United Kingdom

Tom Chen
Department of Electrical Engineering
Colorado State University
Fort Collins, Colorado

James G. Cottle
Hewlett-Packard
San Francisco, California

Yariv Ephraim
Department of Electrical and Computer Engineering
George Mason University
Fairfax, Virginia

Eugene D. Fabricius
EL/EE Department
California Polytechnic State University
San Luis Obispo, California

Robert J. Feugate, Jr.
College of Engineering and Technology
University of Arizona
Flagstaff, Arizona

Paul D. Franzon
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, North Carolina

Susan A. Garrod
Department of Electrical Engineering
Purdue University
West Lafayette, Indiana

James E. Goldman
Purdue University
West Lafayette, Indiana

Margaret H. Hamilton
Hamilton Technologies, Inc.
Cambridge, Massachusetts

Rangachar Kasturi
Department of Computer Science
Pennsylvania State University
State College, Pennsylvania

David A. Kosiba
Pennsylvania State University
State College, Pennsylvania

Paul P.K. Lee
Microelectronics Technical Division
Eastman Kodak Company
Rochester, New York

Élvio João Leonardo
University of Campinas
São Paulo, Brazil

Honoch Lev-Ari
Department of Electrical and Computer Engineering
Northeastern University
Boston, Massachusetts

Shih-Lien Lu
Department of Electronics and Computer Engineering
Oregon State University
Corvallis, Oregon

© 2006 by Taylor & Francis Group, LLC

Álvaro Augusto Machado Medeiros
University of Campinas
São Paulo, Brazil

Victor Meeldijk
Network Processing Group
Intel Corporation
Parsippany, New Jersey

John D. Meyer
Printing Technologies Department
Hewlett-Packard Co.
Palo Alto, California

Wayne Needham
Intel Corporation
Chandler, Arizona

Fabrizio Pollara
Jet Propulsion Lab
California Institute of Technology
Pasadena, California

William J.J. Roberts
Atlantic Coast Technologies, Inc.
Silver Spring, Maryland

Joy S. Shetler
Computer Engineering Program
California Polytechnic State University
San Luis Obispo, California

Sidney Soclof
California State University
San Gabriel, California

Sawasd Tantaratana
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, Massachusetts

Stuart K. Tewksbury
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey

Jerry C. Whitaker
Advance Television Systems Committee
Washington, DC

Bogdan M. Wilamowski
Department of Electrical and Computer Engineering
Auburn University
Auburn, Alabama

Michel D. Yacoub
University of Campinas
Pathumthaui, Brazil

© 2006 by Taylor & Francis Group, LLC

Contents

1 Semiconductor Materials
Stuart K. Tewksbury 1-1

2 Thermal Properties
David F. Besch 2-1

3 Semiconductors
Sidney Soclof 3-1

4 Metal-Oxide-Semiconductor Field-Effect Transistor
John R. Brews 4-1

5 Integrated Circuits
Tom Chen 5-1

6 Integrated Circuit Design
Samuel O. Agbo and Eugene D. Fabricius 6-1

7 Digital Logic Families
Robert J. Feugate, Jr. 7-1

8 Memory Devices
Shih-Lien Lu 8-1

9 Microprocessors
James G. Cottle 9-1

10 D/A and A/D Converters
Susan A. Garrod 10-1

11 Application-Specific Integrated Circuits
Constantine N. Anagnostopoulos and Paul P.K. Lee 11-1

12 Digital Filters
Jonathon A. Chambers, Sawasd Tantaratana and Bruce W. Bomar 12-1

13 Multichip Module Technology
Paul D. Franzon 13-1

© 2006 by Taylor & Francis Group, LLC

14 Testing of Integrated Circuits
Wayne Needham **14**-1

15 Semiconductor Failure Modes
Victor Meeldijk **15**-1

16 Fundamental Computer Architecture
Joy S. Shetler **16**-1

17 Software Design and Development
Margaret H. Hamilton **17**-1

18 Neural Networks and Fuzzy Systems
Bogdan M. Wilamowski **18**-1

19 Machine Vision
David A. Kosiba and Rangachar Kasturi **19**-1

20 A Brief Survey of Speech Enhancement
Yariv Ephraim, Hanoch Lev-Ari and William J.J. Roberts **20**-1

21 Ad Hoc Networks
Michel D. Yacoub, Paulo Cardieri, Élvio João Leonardo, Álvaro Augusto Machado Medeiros **21**-1

22 Network Communication
James E. Goldman **22**-1

23 Printing Technologies and Systems
John D. Meyer **23**-1

24 Data Storage Systems
Jerry C. Whitaker **24**-1

25 Optical Storage Systems
Praveen Asthana **25**-1

26 Error Correction
Fabrizio Pollara **26**-1

© 2006 by Taylor & Francis Group, LLC

1

Semiconductor Materials

1.1 Introduction **1**-1

1.2 Crystalline Structures **1**-2
Basic Semiconductor Materials Groups • Three-Dimensional Crystal Lattice • Crystal Directions and Planes

1.3 Energy Bands and Related Semiconductor Parameters. **1**-6
Conduction and Valence Band • Direct Gap and Indirect Gap Semiconductors • Effective Masses of Carriers • Intrinsic Carrier Densities • Substitutional Dopants

1.4 Carrier Transport **1**-14
Low Field Mobilities • Saturated Carrier Velocities

1.5 Crystalline Defects **1**-18
Point Defects • Line Defects • Stacking Faults and Grain Boundaries • Unintentional Impurities • Surface Defects: The Reconstructed Surface

1.6 Summary **1**-21

Stuart K. Tewksbury

1.1 Introduction

A semiconductor material has a resistivity lying between that of a conductor and that of an insulator. In contrast to the granular materials used for resistors, however, a semiconductor establishes its conduction properties through a complex quantum mechanical behavior within a periodic array of semiconductor atoms, that is, within a crystalline structure. For appropriate atomic elements, the crystalline structure leads to a disallowed energy band between the energy level of electrons bound to the crystal's atoms and the energy level of electrons free to move within the crystalline structure (i.e., not bound to an atom). This energy gap fundamentally impacts the mechanisms through which electrons associated with the crystal's atoms can become free and serve as conduction electrons. The resistivity of a semiconductor is proportional to the free carrier density, and that density can be changed over a wide range by replacing a very small portion (about 1 in 10^6) of the base crystal's atoms with different atomic species (*doping atoms*). The majority carrier density is largely pinned to the net dopant impurity density. By selectively changing the crystalline atoms within small regions of the crystal, a vast number of small regions of the crystal can be given different conductivities. In addition, some dopants establish the electron carrier density (*free electron density*), whereas others establish the *hole* carrier density (holes are the dual of electrons within semiconductors). In this manner, different types of semiconductor (n type with much higher electron carrier density than the hole density and p type with much higher hole carrier density than the electron carrier density) can be located in small but contacting regions within the crystal.

By applying electric fields appropriately, small regions of the semiconductor can be placed in a state in which all of the carriers (electron and hole) have been expelled by the electric field and that electric field

© 2006 by Taylor & Francis Group, LLC

sustained by the exposed dopant ions. This allows electric switching between a conducting state (with a settable resistivity) and a nonconducting state (with conductance vanishing as the carriers vanish).

This combination of localized regions with precisely controlled resistivity (dominated by *electron conduction* or by *hole conduction*) combined with the ability to electronically control the flow of the carriers (electrons and holes) leads to the semiconductors being the foundation for contemporary electronics. This foundation is particularly strong because a wide variety of atomic elements (and mixtures of atomic elements) can be used to tailor the semiconductor material to specific needs. The dominance of silicon semiconductor material in the electronics area (e.g., the very large-scale integrated (VLSI) digital electronics area) contrasts with the rich variety of semiconductor materials widely used in optoelectronics. In the latter case, the ability to adjust the **bandgap** to desired wavelengths of light has stimulated a vast number of optoelectronic components, based on a variety of technologies. Electronic components also provide a role for nonsilicon semiconductor technologies, particularly for very high bandwidth circuits that can take advantage of the higher speed capabilities of semiconductors using atomic elements similar to those used in optoelectronics. This rich interest in nonsilicon technologies will undoubtedly continue to grow, due to the rapidly advancing applications of optoelectronics, for the simple reason that silicon is not suitable for producing an efficient optical source.

1.2 Crystalline Structures

Basic Semiconductor Materials Groups

Most semiconductor materials are crystals created by atomic bonds through which the **valence band** of the atoms are filled with eight electrons through sharing of an electron from each of four nearest neighbor atoms. These materials include semiconductors composed of a single atomic species, with the basic atom having four electrons in its valence band (supplemented by covalent bonds to four neighboring atoms to complete the valence band). These elemental semiconductors, therefore, use atoms from group IV of the atomic chart. Other semiconductor materials are composed of two atoms, one from group N ($N < 4$) and the other from group M ($M > 4$) with $N + M = 8$, filling the valence bands with eight electrons. The major categories of semiconductor material are summarized in the following sections.

Elemental (IV–IV) Semiconductors

Elemental semiconductors consist of crystals composed of only a single atomic element from group IV of the periodic chart, that is, germanium (Ge), silicon (Si), carbon (C), and tin (Sn). Silicon is the most commonly used electronic semiconductor material and is also the most common element on Earth. Table 1.1 summarizes the naturally occurring abundance of some elements used for semiconductors, including nonelemental (compound) semiconductors.

Figure 1.1(a) illustrates the covalent bonding (sharing of outer shell, valence band electrons by two atoms) through which each group IV atom of the crystal is bonded to four neighboring group IV atoms, creating filled outer electron bands of eight electrons.

TABLE 1.1 Abundance (Fraction of Elements Occurring on Earth) of Common Elements Used for Semiconductors

Element	Abundance
Si	0.28
Ga	1.5×10^{-5}
As	1.8×10^{-6}
Ge	5×10^{-6}
Cd	2×10^{-7}
In	1×10^{-7}

© 2006 by Taylor & Francis Group, LLC

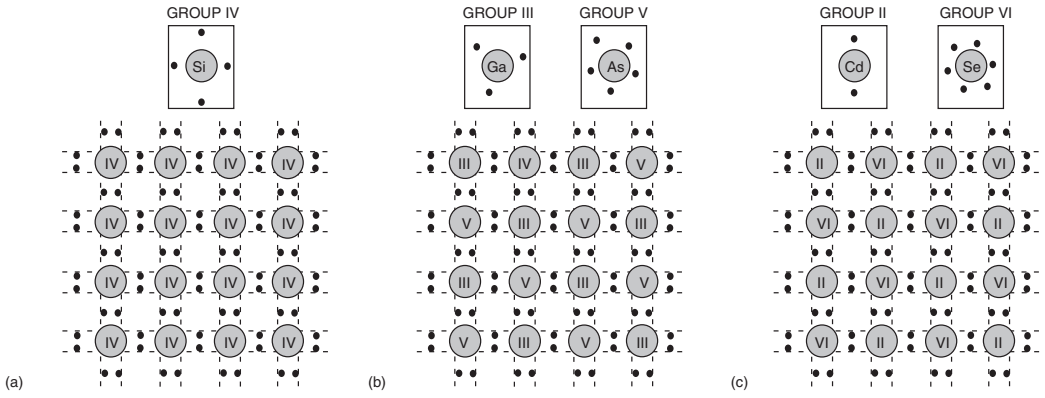


FIGURE 1.1 Bonding arrangements of atoms in semiconductor crystals: (a) elemental semiconductor such as silicon, (b) compound III–V semiconductor such as GaAs, (c) compound II–VI semiconductor such as CdS.

In addition to crystals composed of only a single group IV atomic species, one can also create semiconductor crystals consisting of two or more atoms, all from group IV. For example, silicon carbide (SiC) has been investigated for high-temperature applications. Si_xGe_{1-x} semiconductors are presently under study to achieve bandgap engineering within the silicon system. In this case, a fraction x ($0 < x < 1$) of the atoms in an otherwise silicon crystal is silicon whereas a fraction $1 - x$ has been replaced by germanium. This ability to replace a single atomic element with a combination of two atomic elements from the same column of the periodic chart appears in the other categories of semiconductor described subsequently (and is particularly important for optoelectronic devices).

Compound III–V Semiconductors

The III–V semiconductors are prominent (and will gain in importance) for applications of optoelectronics. In addition, III–V semiconductors have a potential for higher speed operation than silicon semiconductors in electronics applications, with particular importance for areas such as wireless communications. The **compound semiconductors** have a crystal lattice constructed from atomic elements in different groups of the periodic chart. The III–V semiconductors are based on an atomic element A from group III and an atomic element B from group V. Each group III atom is bound to four group V atoms, and each group V atom is bound to four group III atoms, giving the general arrangement shown in Fig. 1.1(b). The bonds are produced by sharing of electrons such that each atom has a filled (eight electron) valence band. The bonding is largely covalent, though the shift of valence charge from the group V atoms to the group III atoms induces a component of ionic bonding to the crystal (in contrast to the elemental semiconductors that have purely covalent bonds). Representative III–V compound semiconductors are GaP, GaAs, GaSb, InP, InAs, and InSb.

GaAs is probably the most familiar example of III–V compound semiconductors, used for both high-speed electronics and for optoelectronic devices. Optoelectronics has taken advantage of ternary and quaternary III–V semiconductors to establish optical wavelengths and to achieve a variety of novel device structures. The **ternary semiconductors** have the general form $(A_x, A'_{1-x})B$ (with two group III atoms used to fill the group III atom positions in the lattice) or $A(B_x, B'_{1-x})$ (using two group V atoms in the group V atomic positions in the lattice). The **quaternary semiconductors** use two group III atomic elements and two group V atomic elements, yielding the general form $(A_x, A'_{1-x})(B_y, B'_{1-y})$. In such constructions, $0 \leq x \leq 1$. Such ternary and quaternary versions are important since the mixing factors (x and y) allow the bandgap to be adjusted to lie between the bandgaps of the simple compound crystals with only one type of group III and one type of group V atomic element. The adjustment of wavelength allows the material to be tailored for particular optical wavelengths, since the

© 2006 by Taylor & Francis Group, LLC

TABLE 1.2 Semiconductor Optical Sources and Representative Wavelengths

Material Layers Used	Wavelength, nm
ZnS	454
AlGaInP/GaAs	580
AlGaAs/GaAs	680
GaInAsP/InP	1580
InGaAsSb/GaSb	2200
AlGaSb/InAsSb/GaSb	3900
PbSnTe/PbTe	6000

wavelength λ of light is related to energy (in this case the gap energy E_g) by $\lambda = hc/E_g$, where h is Planck's constant and c is the speed of light. Table 1.2 provides examples of semiconductor laser materials and a representative optical wavelength for each, providing a hint of the vast range of combinations that are available for optoelectronic applications. Table 1.3, on the other hand, illustrates the change in wavelength (here corresponding to color in the visible spectrum) by adjusting the mixture of a ternary semiconductor.

In contrast to single element elemental semiconductors (for which the positioning of each atom on a lattice site is not relevant), III–V semiconductors require very good control of stoichiometry (i.e., the ratio of the two atomic species) during crystal growth. For example, each Ga atom must reside on a Ga (and not an As) site and vice versa. For these and other reasons, large III–V crystals of high quality are generally more difficult to grow than a large crystal of an elemental semiconductor such as Si.

Compound II–VI Semiconductors

These semiconductors are based on one atomic element from group II and one atomic element from group VI, each type being bonded to four nearest neighbors of the other type, as shown in Fig. 1.1(c). The increased amount of charge from group VI to group II atoms tends to cause the bonding to be more ionic than in the case of III–V semiconductors. II–VI semiconductors can be created in ternary and quaternary forms, much like the III–V semiconductors. Although less common than the III–V semiconductors, the II–VI semiconductors have served the needs of several important applications. Representative II–VI semiconductors are ZnS, ZnSe, and ZnTe (which form in the zinc blende lattice structure discussed subsequently); CdS and CdSe (which can form in either the zinc blende or the wurtzite lattice structure); and CdTe (which forms in the wurtzite lattice structure).

Three-Dimensional Crystal Lattice

The two-dimensional views illustrated in the preceding section provide a simple view of the sharing of valence band electrons and the bonds between atoms. The full three-dimensional lattice structure, however, is considerably more complex than this simple two-dimensional illustration. Fortunately, most semiconductor crystals share a common basic structure, developed as follows.

TABLE 1.3 Variation of x to Adjust Wavelength in $GaAs_xP_{1-x}$ Semiconductors

Ternary Compound	Color
$GaAs_{0.14}P_{0.86}$	Yellow
$GaAs_{0.35}P_{0.65}$	Orange
$GaAs_{0.6}P_{0.4}$	Red

© 2006 by Taylor & Francis Group, LLC

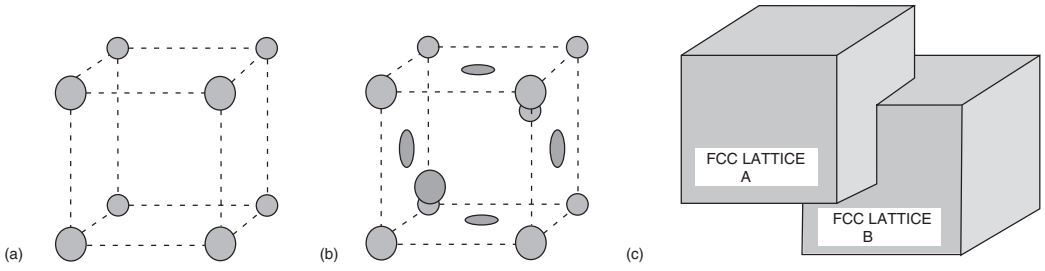


FIGURE 1.2 Three-dimensional crystal lattice structure: (a) basic cubic lattice, (b) face-centered cubic (fcc) lattice, (c) two interpenetrating fcc lattices. In this figure, the dashed lines between atoms are not atomic bonds but, instead, are used merely to show the basic outline of the cube.

The crystal structure begins with a cubic arrangement of eight atoms as shown in Fig. 1.2(a). This cubic lattice is extended to a *face-centered cubic* (fcc) lattice, shown in Fig. 1.2(b), by adding an atom to the center of each face of the cube (leading to a lattice with 14 atoms). The *lattice constant* is the side dimension of this cube.

The full lattice structure combines two of these fcc lattices, one lattice interpenetrating the other (i.e., the corner of one cube is positioned within the interior of the other cube, with the faces remaining parallel), as illustrated in Fig. 1.2(c). For the III–V and II–VI semiconductors with this fcc lattice foundation, one fcc lattice is constructed from one type of element (e.g., type III) and the second fcc lattice is constructed from the other type of element (e.g., group V). In the case of ternary and quaternary semiconductors, elements from the same atomic group are placed on the same fcc lattice. All bonds between atoms occur between atoms in different fcc lattices. For example, all Ga atoms in the GaAs crystal are located on one of the fcc lattices and are bonded to As atoms, all of which appear on the second fcc lattice. The interatomic distances between neighboring atoms is, therefore, less than the lattice constant. For example, the interatomic spacing of Si atoms is 2.35 Å but the lattice constant of Si is 5.43 Å.

If the two fcc lattices contain elements from different groups of the periodic chart, the overall crystal structure is called the *zinc blende* lattice. In the case of an elemental semiconductor such as silicon, silicon atoms appear in both fcc lattices, and the overall crystal structure is called the *diamond* lattice (carbon crystallizes into a diamond lattice creating true diamonds, and carbon is a group IV element). As in the discussion regarding III–V semiconductors, the bonds between silicon atoms in the silicon crystal extend between fcc sublattices.

Although the common semiconductor materials share this basic diamond/zinc blende lattice structure, some semiconductor crystals are based on a hexagonal close-packed (hcp) lattice. Examples are CdS and CdSe. In this example, all of the Cd atoms are located on one hcp lattice whereas the other atom (S or Se) is located on a second hcp lattice. In the spirit of the diamond and zinc blende lattices, the complete lattice is constructed by interpenetrating these two hcp lattices. The overall crystal structure is called a *wurtzite* lattice. Type IV–VI semiconductors (PbS, PbSe, PbTe, and SnTe) exhibit a narrow bandgap and have been used for infrared detectors. The lattice structure of these example IV–VI semiconductors is the simple cubic lattice (also called an NaCl lattice).

Crystal Directions and Planes

Crystallographic directions and planes are important in both the characteristics and the applications of semiconductor materials since different crystallographic planes can exhibit significantly different physical properties. For example, the surface density of atoms (atoms per square centimeter) can differ substantially on different crystal planes. A standardized notation (the so-called *Miller indices*) is used to define the crystallographic planes and directions normal to those planes.

The general crystal lattice defines a set of unit vectors ($\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$) such that an entire crystal can be developed by copying the unit cell of the crystal and duplicating it at integer offsets along the unit vectors,

© 2006 by Taylor & Francis Group, LLC

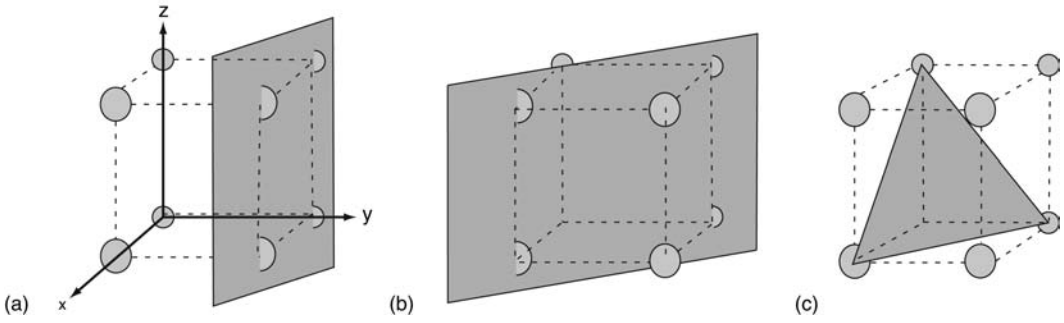


FIGURE 1.3 Examples of crystallographic planes within a cubic lattice organized semiconductor crystal: (a) (010) plane, (b) (110) plane, (c) (111) plane.

that is, replicating the basis cell at positions $n_a\boldsymbol{a} + n_b\boldsymbol{b} + n_c\boldsymbol{c}$, where n_a, n_b, and n_c are integers. The unit vectors need not be orthogonal in general. For the cubic foundation of the diamond and zinc blende structures, however, the unit vectors are in the orthogonal x, y, and z directions.

Figure 1.3 shows a cubic crystal, with basis vectors in the x, y, and z directions. Superimposed on this lattice are three planes (Fig. 1.3(a), Fig 1.3(b), and Fig. 1.3(c)). The planes are defined relative to the crystal axes by a set of three integers (h, k, l) where h corresponds to the plane's intercept with the x axis, k corresponds to the plane's intercept with the y axis, and l corresponds to the plane's intercept with the z axis. Since parallel planes are equivalent planes, the intercept integers are reduced to the set of the three smallest integers having the same ratios as the described intercepts. The (100), (010), and (001) planes correspond to the faces of the cube. The (111) plane is tilted with respect to the cube faces, intercepting the x, y, and z axes at 1, 1, and 1, respectively. In the case of a negative axis intercept, the corresponding Miller index is given as an integer and a bar over the integer, for example, $(\bar{1}00)$, that is, similar to the (100) plane but intersecting the x axis at -1.

Additional notation is used to represent sets of planes with equivalent symmetry and to represent directions. For example, {100} represents the set of equivalent planes (100), $(\bar{1}00)$, (010), $(0\bar{1}0)$, (001), and $(00\bar{1})$. The direction normal to the (hkl) plane is designated $[hkl]$. The different planes exhibit different behavior during device fabrication and impact electrical device performance differently. One difference is due to the different reconstructions of the crystal lattice near a surface to minimize energy. Another is the different surface density of atoms on different crystallographic planes. For example, in Si the (100), (110), and (111) planes have surface atom densities (atoms per square centimeter) of 6.78×10^{14}, 9.59×10^{14}, and 7.83×10^{14}, respectively.

1.3 Energy Bands and Related Semiconductor Parameters

A semiconductor crystal establishes a periodic arrangement of atoms, leading to a periodic spatial variation of the potential energy throughout the crystal. Since that potential energy varies significantly over interatomic distances, quantum mechanics must be used as the basis for allowed energy levels and other properties related to the semiconductor. Different semiconductor crystals (with their different atomic elements and different interatomic spacings) lead to different characteristics. The periodicity of the potential variations, however, leads to several powerful general results applicable to all semiconductor crystals. Given these general characteristics, the different semiconductor materials exhibit properties related to the variables associated with these general results. A coherent discussion of these quantum mechanical results is beyond the scope of this chapter and, therefore, we take those general results as given.

In the case of materials that are semiconductors, a central result is the energy-momentum functions defining the state of the electronic charge carriers. In addition to the familiar electrons, semiconductors also provide holes (i.e., positively charged particles) that behave similarly to the electrons. Two energy levels are important: one is the energy level (**conduction band**) corresponding to electrons that are not

© 2006 by Taylor & Francis Group, LLC

bound to crystal atoms and that can move through the crystal, and the other energy level (valence band) corresponds to holes that can move through the crystal. Between these two energy levels, there is a region of forbidden energies (i.e., energies for which a free carrier cannot exist). The separation between the conduction and valence band minima is called the *energy gap* or bandgap. The energy bands and the energy gap are fundamentally important features of the semiconductor material.

Conduction and Valence Band

In quantum mechanics, a particle is represented by a collection of plane waves ($e^{j(\omega t - \mathbf{k}\cdot\mathbf{x})}$) where the frequency ω is related to the energy E according to $E = \hbar\omega$ and the momentum p is related to the wave vector by $\boldsymbol{p} = \hbar\boldsymbol{k}$. In the case of a classical particle with mass m moving in free space, the energy and momentum are related by $E = p^2/(2m)$ that, using the relationship between momentum and wave vector, can be expressed as $E = (\hbar k)^2/(2m)$. In the case of the semiconductor, we are interested in the energy/momentum relationship for a free electron (or hole) moving in the semiconductor, rather than moving in free space. In general, this E–k relationship will be quite complex, and there will be a multiplicity of E–k states resulting from the quantum mechanical effects. One consequence of the periodicity of the crystal's atom sites is a periodicity in the wave vector $\boldsymbol{k}$, requiring that we consider only values of k over a limited range (with the E–k relationship periodic in k).

Figure 1.4 illustrates a simple example (not a real case) of a conduction band and a valence band in the energy-momentum plane (i.e., the E vs. k plane). The E vs. k relationship of the conduction band will exhibit a minimum energy value and, under equilibrium conditions, the electrons will favor being in that minimum energy state. Electron energy levels above this minimum (E_c) exist, with a corresponding value of momentum. The E vs. k relationship for the valence band corresponds to the energy–momentum relationship for holes. In this case, the energy values increase in the direction toward the bottom of the page and the minimum valence band energy level E_v is the maximum value in Fig. 1.4. When an electron bound to an atom is provided with sufficient energy to become a free electron, a hole is left behind. Therefore,

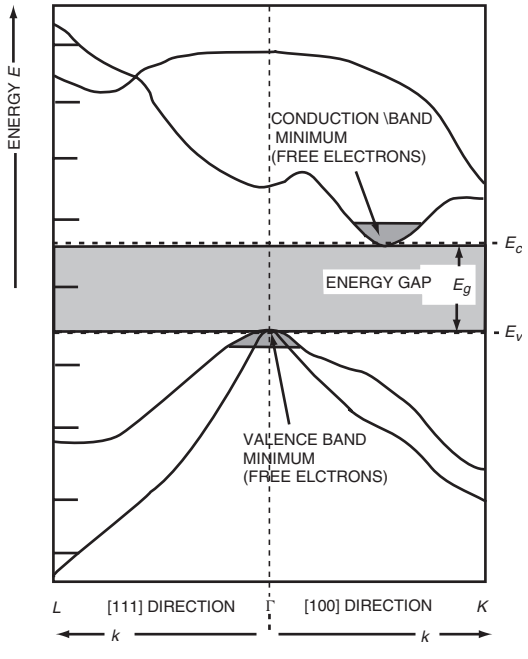


FIGURE 1.4 General structure of conduction and valence bands.

© 2006 by Taylor & Francis Group, LLC

TABLE 1.4 Variation of Energy Gap with Temperature and Pressure

Semiconductor	E_g, 300 K	dE_g/dT, meV/K	dE_g/dP, meV/kbar
Si	1.110	−0.28	−1.41
Ge	0.664	−0.37	5.1
GaP	2.272	−0.37	10.5
GaAs	1.411	−0.39	11.3
GaSb	0.70	−0.37	14.5
InP	1.34	−0.29	9.1
InAs	0.356	−0.34	10.0
InSb	0.180	−0.28	15.7
ZnSe	2.713	−0.45	0.7
ZnTe	2.26	−0.52	8.3
CdS	2.485	−0.41	4.5
CdSe	1.751	0.36	5
CdTe	1.43	−0.54	8

Source: Adapted from Böer, K.W. 1990. *Survey of Semiconductor Physics, Vol. 1: Electrons and Other Particles in Bulk Semiconductors*. Van Nostrand, New York.

the energy gap $E_g = E_c - E_v$ represents the minimum energy necessary to generate an electron-hole pair (higher energies will initially produce electrons with energy greater than E_c, but such electrons will generally lose energy and fall into the potential minimum).

The details of the energy bands and the bandgap depend on the detailed quantum mechanical solutions for the semiconductor crystal structure. Changes in that structure (even for a given semiconductor crystal such as Si) can therefore lead to changes in the energy band results. Since the thermal coefficient of expansion of semiconductors is nonzero, the bandgap depends on temperature due to changes in atomic spacing with changing temperature. Changes in pressure also lead to changes in atomic spacing. Though these changes are small, the are observable in the value of the energy gap. Table 1.4 gives the room temperature value of the energy gap E_g for several common semiconductors, along with the rate of change of E_g with temperature (T) and pressure (P) at room temperature.

The temperature dependence, though small, can have a significant impact on carrier densities. A heuristic model of the temperature dependence of E_g is $E_g(T) = E_g(0\,\text{K}) - \alpha T^2/(T+\beta)$. Values for the parameters in this equation are provided in Table 1.5. Between 0 and 1000 K, the values predicted by this equation for the energy gap of GaAs are accurate to about 2×10^{-3} eV.

Direct Gap and Indirect Gap Semiconductors

Figure 1.5 illustrates the energy bands for Ge, Si, and GaAs crystals. In Fig. 1.5(b), for silicon, the valence band has a minimum at a value of $\boldsymbol{k}$ different than that for the conduction band minimum. This is an **indirect gap**, with generation of an electron-hole pair requiring an energy E_g and a change in momentum (i.e., k). For direct recombination of an electron-hole pair, a change in momentum is also required. This requirement for a momentum change (in combination with energy and momentum conservation laws) leads to a

TABLE 1.5 Temperature Dependence Parameters for Common Semiconductors

	E_g (0 K), eV	$\alpha(\times 10^{-4})$	β	E_g(300 K), eV
GaAs	1.519	5.405	204	1.42
Si	1.170	4.73	636	1.12
Ge	0.7437	4.774	235	0.66

Source: Adapted from Sze, S.M. 1981. *Physics of Semiconductor Devices*, 2nd ed. Wiley-Interscience, New York.

© 2006 by Taylor & Francis Group, LLC

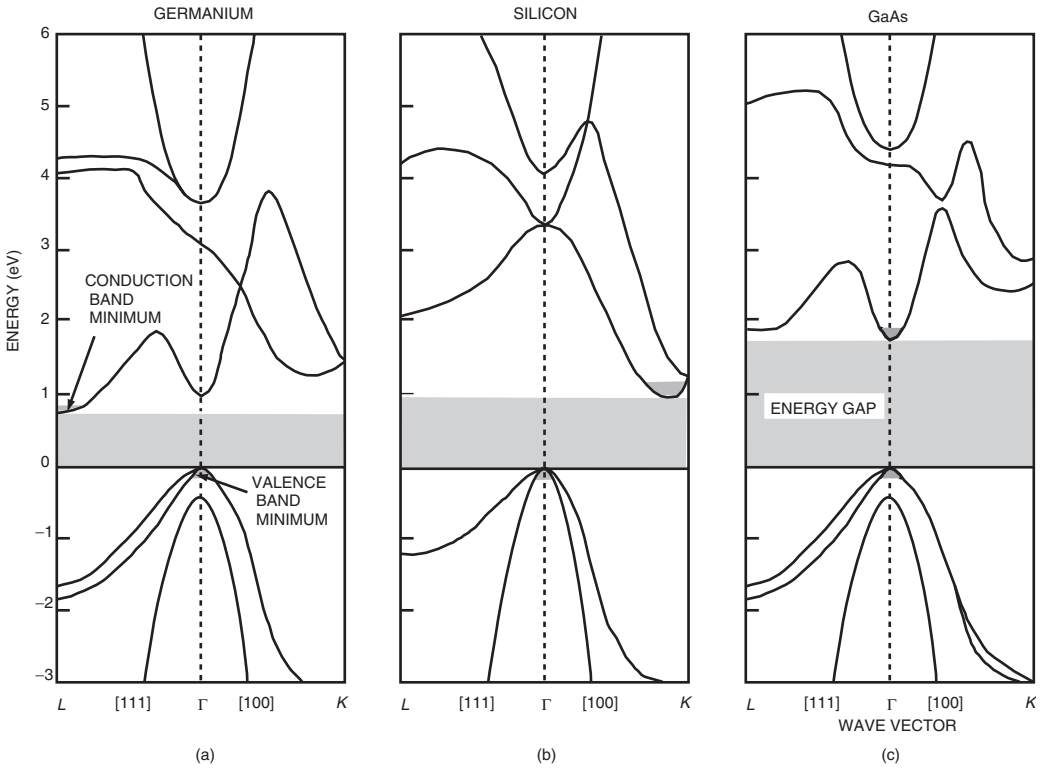


FIGURE 1.5 Conduction and valence bands for (a) Ge, (b) Si, (c) GaAs. (*Source:* Adapted from Sze, S.M. 1981. *Physics of Semiconductor Devices*, 2nd ed. Wiley Interscience, New York.)

requirement that a phonon participate with the carrier pair during a direct recombination process generating a photon. This is a highly unlikely event, rendering silicon ineffective as an optoelectronic source of light. The direct generation process is more readily allowed (with the simultaneous generation of an electron, a hole, and a phonon), allowing silicon and other **direct gap semiconductors** to serve as optical detectors.

In Fig. 1.5(c), for GaAs, the conduction band minimum and the valence band minimum occur at the same value of momentum, corresponding to a direct gap. Since no momentum change is necessary during direct recombination, such recombination proceeds readily, producing a photon with the energy of the initial electron and hole (i.e., a photon energy equal to the bandgap energy). For this reason, direct gap semiconductors are efficient sources of light (and use of different direct gap semiconductors with different E_g provides a means of tailoring the wavelength of the source). The wavelength λ corresponding to the gap energy is $\lambda = hc/E_g$.

Figure 1.5(c) also illustrates a second conduction band minimum with an indirect gap, but at a higher energy than the minimum associated with the direct gap. The higher conduction band minimum can be populated by electrons (which are in an equilibrium state of higher energy) but the population will decrease as the electrons gain energy sufficient to overcome that upper barrier.

Effective Masses of Carriers

For an electron with energy close to the minimum of the conduction band, the energy vs. momentum relationship along each momentum axis is approximately given by $E(k) = E_0 + a_2(k - k^*)^2 + a_4(k - k^*)^4 + \cdots$. Here, $E_0 = E_c$ is the *ground state energy* corresponding to a free electron at rest and k^* is

© 2006 by Taylor & Francis Group, LLC

the wave vector at which the conduction band minimum occurs. Only even powers of $k - k^*$ appear in the expansion of $E(k)$ around k^* due to the symmetry of the E–k relationship around $k = k^*$. This approximation holds for sufficiently small increases in E above E_c. For sufficiently small movements away from the minimum (i.e., sufficiently small $k - k^*$), the terms in $k - k^*$ higher than quadratic can be ignored and $E(k) \approx E_0 + a_2 k^2$, where we have taken $k^* = 0$. If, instead of a free electron moving in the semiconductor crystal, we had a free electron moving in free space with potential energy E_0, the energy-momentum relationship would be $E(k) = E_0 + (\hbar k)^2/(2m_0)$, where m_0 is the mass of an electron. By comparison of these results, it is clear that we can relate the curvature coefficient a_2 associated with the parabolic minimum of the conduction band to an **effective mass** m_e^*, that is, $a_2 = (\hbar^2)/(2m_e^*)$ or

$$\frac{1}{m_e^*} = \frac{2}{\hbar^2} \cdot \frac{\partial^2 E_c(k)}{\partial k^2}$$

Similarly for holes, an effective mass m_h^* of the holes can be defined by the curvature of the valence band minimum, that is,

$$\frac{1}{m_h^*} = \frac{2}{\hbar^2} \cdot \frac{\partial^2 E_v(k)}{\partial k^2}$$

Since the energy bands depend on temperature and pressure, the effective masses can also be expected to have such dependencies, though the room temperature and normal pressure value is normally used in device calculations.

This discussion assumes the simplified case of a scalar variable k. In fact, the wave vector $\boldsymbol{k}$ has three components (k_1, k_2, k_3), with directions defined by the unit vectors of the underlying crystal. Therefore, there are separate masses for each of these vector components of $\boldsymbol{k}$, that is, masses m_1, m_2, m_3. A scalar mass m^* can be defined using these directional masses, the relationship depending on the details of the directional masses. For cubic crystals (as in the diamond and zinc blende structures), the directions are the usual orthonormal directions and $m^* = (m_1 \cdot m_2 \cdot m_3)^{1/3}$. The three directional masses effectively reduce to two components if two values are equal (e.g., $m_1 = m_2$), as in the case of *longitudinal* and *transverse* effective masses (m_l and m_t, respectively) seen in silicon and several other semiconductors. In this case, $m^* = [(m_t)^2 \cdot m_l]^{1/3}$. If all three values of m_1, m_2, m_3 are equal, then a single value m^* can be used.

An additional complication is seen in the valence band structures in Fig. 1.5. Here, two different E–k valence bands have the same minima. Since their curvatures are different, the two bands correspond to different masses, one corresponding to *heavy holes* with mass m_h and the other to *light holes* with mass m_l. The effective scalar mass in this case is $m^* = (m_h^{3/2} + m_l^{3/2})^{2/3}$. Such light and heavy holes occur in several semiconductors, including Si.

Values of effective mass are given in Tables 1.8 and 1.5.

Intrinsic Carrier Densities

The density of free electrons in the conduction band depends on two functions. One is the density of states $D(E)$ in which electrons can exist and the other is the energy distribution function $F(E, T)$ of free electrons.

The energy distribution function (under thermal equilibrium conditions) is given by the Fermi-Dirac distribution function

$$F(E) = \left[1 + \exp\left(\frac{E - E_f}{k_B T}\right)\right]^{-1}$$

which, in most practical cases, can be approximated by the classical Maxwell-Boltzmann distribution. These distribution functions are general functions, not dependent on the specific semiconductor material.

© 2006 by Taylor & Francis Group, LLC

The density of states $D(E)$, on the other hand, depends on the semiconductor material. A common approximation is

$$D_n(E) = M_c \frac{\sqrt{2}}{\pi^2} \frac{(E - E_c)^{1/2}}{\hbar^3} \frac{(m_e^*)^3}{2}$$

for electrons and

$$D_p(E) = M_v \frac{\sqrt{2}}{\pi^2} \frac{(E_v - E)^{1/2}}{\hbar^3} \frac{(m_h^*)^3}{2}$$

for holes. Here, M_c and M_v are the number of equivalent minima in the conduction band and valence band, respectively. Note that, necessarily, $E \geq E_c$ for free electrons and $E \leq E_v$ for free holes due to the forbidden region between E_c and E_v.

The density n of electrons in the conduction band can be calculated as

$$n = \int_{E=E_c}^{\infty} F(E,T)D(E)dE$$

For Fermi levels significantly (more than a few $k_B T$) below E_c and above E_v, this integration leads to the results

$$n = N_c e^{-(E_c - E_f)/k_b T}$$

and

$$p = N_v e^{-(E_f - E_v)/k_b T}$$

where n and p are the densities of free electrons in the conduction band and of holes in the valence band, respectively. N_c and N_v are *effective densities of states* that vary with temperature (slower than the exponential in the preceding equations), effective mass, and other conditions. Table 1.6 gives values of N_c and N_v for several semiconductors. Approximate expressions for these densities of state are $N_c = 2(2\pi m_e^* k_B T/\hbar^2)^{3/2} M_c$ and $N_v = 2(2\pi m_e^* k_B T/\hbar^2)^{3/2} M_v$. These effective densities of states are fundamental parameters used in evaluating the electrical characteristics of semiconductors. The preceding equations for n and p apply both to **intrinsic semiconductors** (i.e., semiconductors with no impurity dopants) as well as to semiconductors

TABLE 1.6 N_c and N_v at 300 K

	N_c ($\times 10^{19}$/cm^3)	N_v ($\times 10^{19}$/cm^3)
Ge	1.54	1.9
Si	2.8	1.02
GaAs	0.043	0.81
GaP	1.83	1.14
GaSb	0.021	0.62
InAs	0.0056	0.62
InP	0.052	1.26
InSb	0.0043	0.62
CdS	0.224	2.5
CdSe	0.11	0.74
CdTe	0.13	0.55
ZnSe	0.31	0.87
ZnTe	0.22	0.078

Source: Adapted from Böer, K.W. 1990. *Survey of Semiconductor Physics, Vol. 1: Electrons and Other Particles in Bulk Semiconductors*. Van Nostrand, New York.

© 2006 by Taylor & Francis Group, LLC

that have been doped with donor and/or acceptor impurities. Changes in the interrelated values of n and p through introduction of dopant impurities can be represented by changes in a single variable, the Fermi level E_f.

The product of n and p is independent of Fermi level and is given by

$$n \cdot p = N_c \cdot N_v e^{-E_g/k_B T}$$

where the energy gap $E_g = E_c - E_v$. Again, this holds for both intrinsic semiconductors and for doped semiconductors. In the case of an intrinsic semiconductor, charge neutrality requires that $n = p \equiv n_i$, where n_i is the *intrinsic carrier concentration* and

$$n_i^2 = N_c \cdot N_v e^{-E_g/k_B T}$$

Since, under thermal equilibrium conditions $np \equiv n_i^2$ (even under impurity doping conditions), knowledge of the density of one of the carrier types (e.g., of p) allows direct determination of the density of the other (e.g., $n = n_i^2/p$). Values of n_i vary considerably among semiconductor materials: $2 \times 10^{-3}/\text{cm}^3$ for CdS, $3.3 \times 10^6/\text{cm}^3$ for GaAs, $0.9 \times 10^{10}/\text{cm}^3$ for Si, $1.9 \times 10^{13}/\text{cm}^3$ for Ge, and 9.1×10^{14} for PbS.

Since there is appreciable temperature dependence in the effective density of states, the equations do not accurately represent the temperature variations in n_i over wide temperature ranges. Using the approximate expressions for N_c and N_v already given,

$$n_i = 2\left(2\pi k_B T/\hbar^2\right)^{3/2}\left(m_e^* m_h^*\right)^{3/4}\sqrt{M_c M_v}\, e^{-E_g/k_B T}$$

exhibiting a $T^{3/2}$ temperature dependence superimposed on the exponential dependence on $1/T$. For example, at 300 K

$$n_i(T) = 1.76 \times 10^{16} T^{3/2} e^{-4550/T}\ \text{cm}^{-3} \quad \text{for Ge}$$

and

$$n_i(T) = 3.88 \times 10^{16} T^{3/2} e^{-7000/T}\ \text{cm}^{-3} \quad \text{for Si}$$

Substitutional Dopants

An intrinsic semiconductor material contains only the elemental atoms of the basic material (e.g., silicon atoms for Si, gallium and arsenic atoms for GaAs, etc.). The resistivity is quite high for such intrinsic semiconductors and doping is used to establish a controlled lower resistivity material and to establish pn junctions (interface between p-type and n-type regions of the semiconductor). Doping concentrations are generally in the range 10^{14}–$10^{17}/\text{cm}^3$, small relative to the density of atoms in the crystal (e.g., to the density 5×10^{22} atoms/cm^3 of silicon atoms in Si crystals). Table 1.7 lists a variety of dopants and their energy levels for Si and GaAs.

TABLE 1.7 Acceptor and Donor Impurities Used in Si and GaAs

	Donor	$E_c - E_d$, eV	Acceptor	$E_a - E_v$, eV
Si crystal	Sb	0.039	B	0.045
	P	0.045	Al	0.067
	As	0.054	Ga	0.073
GaAs crystal	S	0.006	Mg	0.028
	Se	0.006	Zn	0.031
	Te	0.03	Cd	0.035
	Si	0.058	Si	0.026

Source: Adapted from Tyagi, M.S. 1991. *Introduction to Semiconductor Materials.* Wiley, New York.

© 2006 by Taylor & Francis Group, LLC

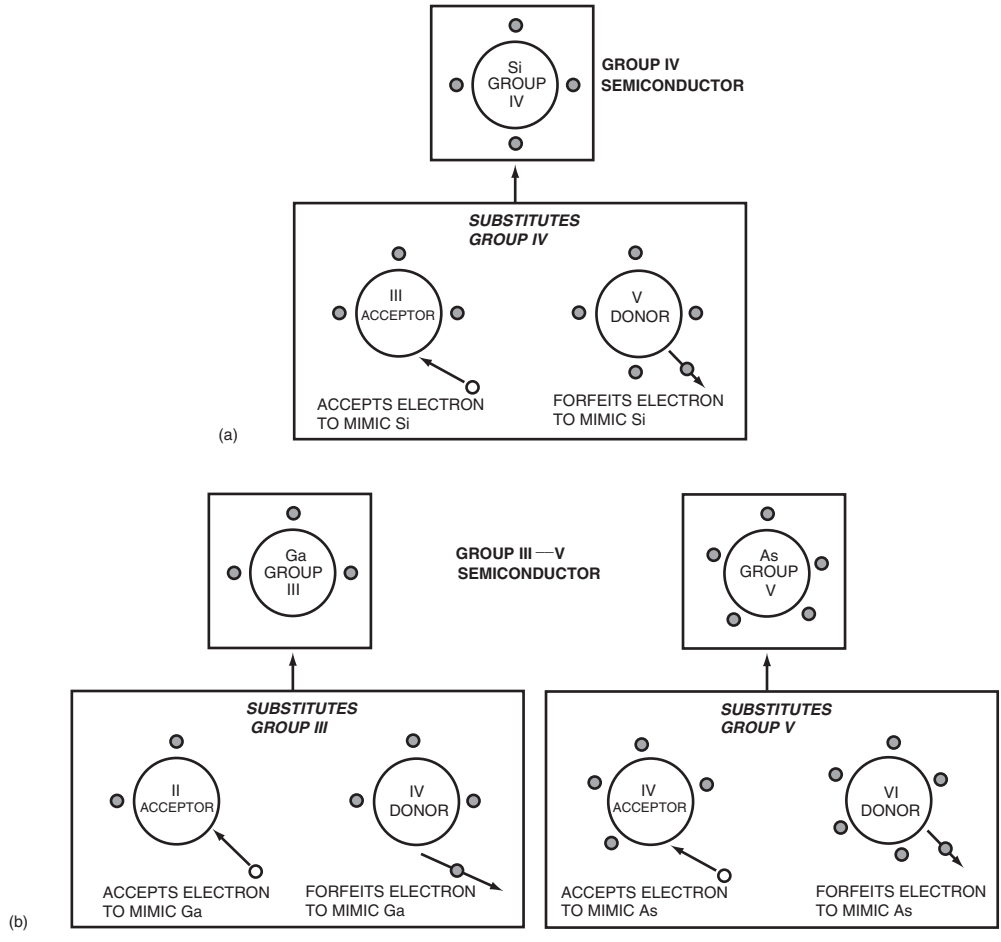


FIGURE 1.6 Substitution of dopant atoms for crystal atoms: (a) IV–IV semiconductors (e.g., silicon), (b) III–V semiconductors (e.g., GaAs).

Figure 1.6(a) illustrates acceptor dopants and donor dopants in silicon. In the case of acceptor dopants, group III elements of the periodic chart are used to substitute for the group IV silicon atoms in the crystal. This acceptor atom has one fewer electron in its outer shell than the silicon atom and readily captures a free electron to provide the missing electron needed to complete the outer shells (eight electrons) provided by the covalent bonds. The acceptor atom, with a captured electron, becomes a negative ion. The electron captured from the outer shell of a neighboring silicon atom leaves behind a hole at that neighboring silicon atom (i.e., generates a free hole when ionized). By using acceptor doping densities N_A substantially greater than n_i, a net hole density $p \gg n_i$ is created. With $np = n_i^2$ a constant, the electron density n decreases as p increases above n_i. The resulting semiconductor material becomes p type. In the case of donor dopants, group V elements are used to substitute for a silicon atom in the crystal. The donor atom has one extra electron in its outer shell, compared to a silicon atom, and that extra electron can leave the donor site and become a free electron. In this case, the donor becomes a positive ion, generating a free electron. By using donor doping densities N_D substantially greater than n_i, a net electron density $n \gg n_i$ is created, and p decreases substantially below n_i to maintain the np product $np = n_i^2$. An n type semiconductor is produced.

Figure 1.6(b) illustrates the alternative doping options for a III–V semiconductor (GaAs used as an example). Replacement of a group III element with a group II element renders that group II element an acceptor (one fewer electron). Replacement of a group V element with a group VI element renders that

© 2006 by Taylor & Francis Group, LLC

group VI element a donor (one extra electron). Group IV elements such as silicon can also be used for doping. In this case, the group IV element is a donor if it replaces a group III element in the crystal and is an acceptor if it replaces a Group V element in the crystal. Impurities which can serve as either an acceptor or as a donor within a crystal are called **amphoteric impurities**.

Acceptor and donor impurities are most effectively used when the energy required to generate a carrier is small. In the case of small ionization energies (in the crystal lattice), the energy levels associated with the impurities lie within the bandgap close to their respective bands (i.e., donor ionization energies close to the conduction band and acceptor ionization energies close to the valence band). If the difference between the ionization level and the corresponding valence/conduction band is less than about $3k_BT$ (≈ 0.075 eV at 300 K), then the impurities (called **shallow energy level dopants**) are essentially fully ionized at room temperature. The dopants listed in Table 1.7 are such shallow energy dopants. A semiconductor doped with $N_A \gg n_i$ acceptors then has a hole density $p \approx N_A$ and a much smaller electron density $n \approx n_i^2/N_A$. Similarly, a semiconductor doped with $N_D \gg n_i$ donors has an electron density $n \approx N_D$ and a much smaller hole density $p = n_i^2/N_D$. From the results given earlier for carrier concentrations as a function of Fermi level, the Fermi level is readily calculated (given the majority carrier concentration).

Most semiconductors can be selectively made (by doping) either n type or p type, in which case they are called **ambipolar semiconductors.** Some semiconductors can be selectively made only n type or only p type. For example, ZnTe is always p type whereas CdS is always n type. Such semiconductors are called **unipolar semiconductors.**

Dopants with energy levels closer to the center of the energy gap (i.e., the so-called **deep energy level dopants**) serve as electron-hole recombination sites, impacting the minority carrier lifetime and the dominant recombination mechanism in indirect bandgap semiconductors.

1.4 Carrier Transport

Currents in semiconductors arise both due to movement of free carriers in the presence of an electric field and due to diffusion of carriers from high, carrier density regions into lower, carrier density regions. Currents due to electric fields are considered first.

In earlier discussions, it was noted that the motion of an electron in a perfect crystal can be represented by a free electron with an effective mass m_e^* somewhat different than the real mass m_e of an electron. In this model, once the effective mass has been determined, the atoms of the perfect crystal can be discarded and the electron viewed as moving within free space. If the crystal is not perfect, however, those deviations from perfection remain after the perfect crystal lattice has been discarded and act as scattering sites within the free space seen by the electron in the crystal.

Substitution of a dopant for an element of the perfect crystal leads to a distortion of the perfect lattice from which electrons can scatter. If that substitutional dopant is ionized, the electric field of that ion adds to the scattering. Impurities located at interstitial sites (i.e., between atoms in the normal lattice sites) also disrupt the perfect crystal and lead to scattering sites. Crystal defects (e.g., a missing atom) disrupt the perfect crystal and appears as a scattering site in the free space seen by the electron. In useful semiconductor crystals, the density of such scattering sites is small relative to the density of silicon atoms. As a result, removal of the silicon atoms through use of the effective mass leaves a somewhat sparsely populated space of scattering sites.

The perfect crystal corresponds to all atomic elements at the lattice positions and not moving, a condition which can occur only at 0 K. At temperatures above absolute zero, the atoms have thermal energy that causes them to move away from their ideal site. As the atom moves away from the nominal, equilibrium site, forces act to return it to that site, establishing the conditions for a classical oscillator problem (with the atom oscillating about its equilibrium location). Such oscillations of an atom can transfer to a neighboring atom (by energy exchange), leading to the oscillation propagating through space. This wavelike disturbance is called a *phonon* and serves as a scattering site (*phonon scattering*, also called *lattice scattering*), which can appear anywhere in the crystal.

© 2006 by Taylor & Francis Group, LLC

Low Field Mobilities

The dominant scattering mechanisms in silicon are ionized impurity scattering and phonon scattering, though other mechanisms such as mentioned previously do contribute. Within this free space contaminated by scattering centers, the free electron moves at a high velocity (the thermal velocity v_{therm}) set by the thermal energy ($k_B T$) of the electron, with $2k_B T/3 = 0.5\, m_e^* v_{\text{therm}}^2$ and, therefore, $v_{\text{therm}} = \sqrt{4k_B T/3m_e^*}$. At room temperature in Si, the thermal velocity is about 1.5×10^7 cm/s, substantially higher than most velocities that will be induced by applied electric fields or by diffusion. Thermal velocities depend reciprocally on effective mass, with semiconductors having lower effective masses displaying higher thermal velocities than semiconductors with higher effective masses. At these high thermal velocities, the electron will have an average mean time τ_n between collisions with the scattering centers during which it is moving as a free electron in free space. It is during this period between collisions that an external field acts on the electron, creating a slight displacement of the orbit of the free electron. Upon colliding, the process starts again, producing again a slight displacement in the orbit of the free electron. This displacement divided by the mean free time τ_n between collisions represents the velocity component induced by the external electric field. In the absence of such an electric field, the electron would be scattered in random directions and display no net displacement in time. With the applied electric field, the electron has a net drift in the direction set by the field. For this reason, the induced velocity component is called the *drift velocity* and the thermal velocities can be ignored.

By using the standard force equation $F = eE = m_e^* dv/dt$ with velocity $v = 0$ at time $t = 0$ and with an acceleration time τ_n, the final velocity v_f after time τ_n is then simply $v_f = e\tau_n E/m_e^*$ and, letting $v_{\text{drift}} = v_f/2$, $v_{\text{drift}} = e\tau_n E/(2m_e^*)$. Table 1.8 gives values of the electron and hole effective masses for semiconductors.

The drift velocity v_{drift} in an external field E is seen to vary as $v_{\text{drift}} = \mu_n E$, where the electron's **low field mobility** μ_n is given approximately by $\mu_n \approx e\tau_n/(2m_e^*)$. Similarly, holes are characterized by a low field mobility $\mu_p \approx e\tau_p/(2m_h^*)$, where τ_p is the mean time between collision for holes.

This simplified mobility model yields a mobility that is inversely proportional to the effective mass. The effective electron masses in GaAs and Si are $0.09m_e$ and $0.26m_e$, respectively, suggesting a higher electron mobility in GaAs than in Si (in fact, the electron mobility in GaAs is about three times greater than that in Si). The electron and hole effective masses in Si are 0.26 and $0.38m_e$, respectively, suggesting a higher electron mobility than hole mobility in Si (in Si, $\mu_n \approx 1400$ cm^2/V · s and $\mu_p \approx 500$ cm^2/V · s). The simplified model for μ_n and μ_p is based on the highly simplified model of the scattering conditions encountered by carriers moving with thermal energy. Far more complex analysis is necessary to obtain theoretical values of these mobilities. For this reason, the approximate model should be regarded as a guide to mobility variations among semiconductors and not as a predictive model.

The linear dependence of the mobility μ_n (μ_p) on τ_n (τ_p), suggested by the simplified development given previously, also provides a qualitative understanding of the mobility dependence on impurity doping and on temperature. As noted earlier, phonon scattering and ionized impurity scattering are the dominant mechanisms controlling the scattering of carriers in most semiconductors. At room temperature and for normally used impurity doping concentrations, phonon scattering typically dominates ionized impurity scattering. As the temperature decreases, the thermal energy of the crystal atoms decreases, leading to a decrease in the phonon scattering and an increase in the mean free time between phonon scattering events. The result is a mobility μ_{phonon} that increases with decreasing temperature according to

TABLE 1.8 Conductivity Effective Masses for Common Semiconductors

	Ge	Si	GaAs
m_e^*	$0.12m_0$	$0.26m_0$	$0.063m_0$
m_h^*	$0.23m_0$	$0.38m_0$	0.53

© 2006 by Taylor & Francis Group, LLC

TABLE 1.9 Mobility and Temperature Dependence at 300 K

	Ge		Si		GaAs	
	μ_n	μ_p	μ_n	μ_p	μ_n	μ_p
Mobility, $cm^{-2}/V \cdot s$	3900	1900	1400	470	8000	340
Temperature	$T^{-1.66}$	$T^{-2.33}$	$T^{-2.5}$	$T^{-2.7}$	—	$T^{-2.3}$

Source: Adapted from Wang, S. 1989. *Fundamentals of Semiconductor Theory and Device Physics*. Prentice-Hall, Englewood Cliffs, NJ.

$\mu_{\text{phonon}} \approx B_1 T^{-\alpha}$ where α is typically between 1.6 and 2.8 at 300 K. Table 1.9 gives the room temperature mobility and temperature dependence of mobility at 300 K for Ge, Si, and GaAs. In the case of ionized impurity scattering, the lower temperature leads to a lower thermal velocity, with the electrons passing ionized impurity sites more slowly and suffering a stronger scattering effect. As a result, the mobility μ_{ion} decreases with decreasing temperature. Starting at a sufficiently high temperature that phonon scattering dominates, the overall mobility will initially increase with decreasing temperature until the ionized impurity scattering becomes dominant, leading to subsequent decreases in the overall mobility with decreasing temperature.

In Si, Ge, and GaAs, for example, the mobility increases by a factor of roughly 7 at 77 K relative to the value at room temperature, illustrating the dominant role of phonon scattering in these semiconductors under normal doping conditions.

Since scattering probabilities for different mechanisms add to yield the net scattering probability that, in turn, defines the overall mean free time between collisions, mobilities (e.g., phonon scattering mobility and lattice scattering mobility) due to different mechanisms are combined as $\mu^{-1} = \mu_{\text{phonon}}^{-1} + \mu_{\text{ion}}^{-1}$, that is, the smallest mobility dominates.

The mobility due to lattice scattering depends on the density of ionized impurity sites, with higher impurity densities leading to shorter distances between scattering sites and smaller mean free times between collisions. For this reason, the mobility shows a strong dependence on impurity doping levels at temperatures for which such scattering is dominant. As the ionized impurity density increases, the temperature at which the overall mobility becomes dominated by impurity scattering increases. In Si at room temperature, for example, $\mu_e \approx 1400$ and $\mu_p \approx 500$ for dopant concentrations below $\approx 10^{15}$ cm^3, decreasing to approximately 300 and 100, respectively, for concentrations $> 10^{18}$ cm^3.

These qualitative statements can be made substantially more quantitative by providing detailed plots of mobility vs. temperature and vs. impurity density for the various semiconductor materials. Examples of such plots are provided in several references (e.g., Tyagi, 1991; Nicollian and Brews, 1982; Shur, 1987; Sze, 1981; Böer, 1990, 1992; Haug, 1975; Wolfe, Holonyak, and Stillman, 1989; Smith, 1978; Howes and Morgan, 1985).

Diffusion results from a gradient in the carrier density. For example, the flux of electrons due to diffusion is given by $F_e = -D_n \mathrm{d}n/\mathrm{d}x$, with $F_p = -D_p \mathrm{d}p/\mathrm{d}x$ for holes. The diffusion constants D_n and D_p are related to the mobilities given by $D_n = \mu_n(k_b T/e)\mu_e$ and $D_p = (k_B T/e)\mu_p$, the so-called *Einstein relations*. In particular, the mean time between collisions t_{col} determines both the mobility and the diffusion constant.

Saturated Carrier Velocities

The mobilities discussed previously are called low field mobilities since they apply only for sufficiently small electric fields. The low field mobilities represent the scattering from distortions of the perfect lattice, with the electron gaining energy from the electric field between collisions at such distortion sites. At sufficiently high electric fields, the electron gains sufficient energy to encounter inelastic collisions with the elemental atoms of the perfect crystal. Since the density of such atoms is very high (i.e., compared to the density of scattering sites), this new mechanism dominates the carrier velocity at sufficiently high fields

© 2006 by Taylor & Francis Group, LLC

TABLE 1.10 Saturated Velocity and Critical Field for Si at Room Temperature

	Saturated Velocity, cm/s	Critical Electric Field, V/cm
Electrons	1.1×10^7	8×10^3
Holes	9.5×10^6	1.95×10^4

Source: Adapted from Tyagi, M.S. 1991. *Introduction to Semiconductor Materials.* Wiley, New York.

and causes the velocity to become independent of the field (i.e., regardless of the electric field strength, the electron accelerates to a peak velocity at which the inelastic collisions appear and the energy of the electron is lost). The electric field at which velocities become saturated is referred to as the *critical field* E_{cr}. Table 1.10 summarizes the **saturated velocities** and critical fields for electrons and holes in Si. The saturated velocities in GaAs and Ge are about 6×10^6 cm/s, slightly lower than the saturated velocity in Si.

Figure 1.7 shows representative velocity vs. electric field characteristics for electrons in silicon and in GaAs. The linear dependence at small fields illustrates the low field mobility, with GaAs having a larger low field mobility than silicon. The saturated velocities v_{sat}, however, do not exhibit such a large difference between Si and GaAs. Also, the saturated velocities do not exhibit as strong a temperature dependence as the low field mobilities (since the saturated velocities are induced by large electric fields, rather than being perturbations on thermal velocities).

Figure 1.7 also exhibits an interesting feature in the case of GaAs. With increasing electric field, the velocity initially increases beyond the saturated velocity value, falling at higher electric fields to the saturated velocity. This *negative differential mobility* region has been discussed extensively as a potential means of achieving device speeds higher than would be obtained with fully saturated velocities. Table 1.11 summarizes the peak velocities for various semiconductors.

As device dimensions have decreased, the saturated velocities have become a more severe limitation. With critical fields in silicon about 10^4 V/cm, one volt across one micron leads to saturated velocities. Rather than achieving a current proportional to applied voltage (as in the case of the low field mobility condition), currents become largely independent of voltage under saturated velocity conditions. In addition, the

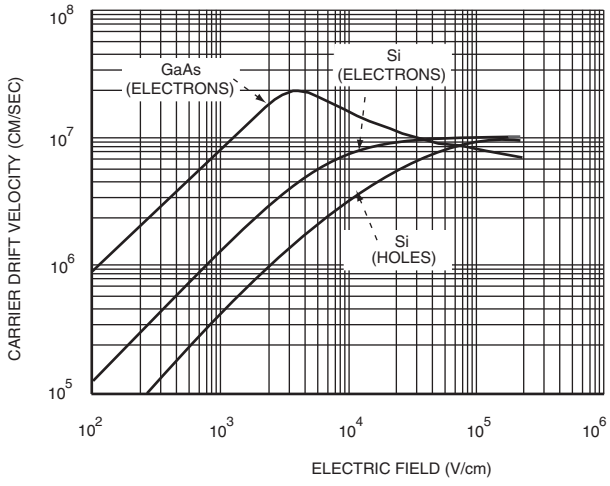


FIGURE 1.7 Velocity vs. electric field for silicon and GaAs semiconductors. (*Source:* Adapted from Sze, S.M. 1981. *Physics of Semiconductor Devices*, 2nd ed. Wiley Interscience, New York with permission.)

© 2006 by Taylor & Francis Group, LLC

TABLE 1.11 Saturated Velocity and Critical Field for Various Semiconductors

Semiconductor	Peak Velocity, cm/s
AsAs	6×10^6
AlSb	7×10^6
GaP	1.2×10^7
GaAs	2×10^7
PbTe	1.7×10^7
InP	4×10^7
InSb	7×10^7

emergence of saturated velocities also compromises the speed advantages implied by high mobilities for some semiconductors (e.g., GaAs vs. Si). In particular, although higher low field velocities generally lead to higher device speeds under low field conditions, the saturated velocities give similar speed performance at high electric fields.

1.5 Crystalline Defects

A variety of defects occur in semiconductor crystals, many of which lead to degradations in performance and require careful growth and fabrication conditions to minimize their occurrence. Other defects are relatively benign. This section summarizes several of the defect types.

Point Defects

The point defects correspond to a lattice atom missing from its position. Two distinct cases appear, as shown in Fig. 1.8. *Schottky defects*, shown in Fig. 1.16(a), result when an atom is missing from a lattice site. Typically the atom is assumed to have migrated to the surface (during crystal growth) where it takes a normal position at a surface lattice site. An energy E_s is required to move a lattice atom to the crystal surface, that energy serves as an activation energy for Schottky defects. At temperature T, the equilibrium concentration N_{sd} of Schottky defects is given by $N_{sd} = N_L \exp(-E_{sd}/kT)$, where N_L is the density of lattice atoms, and there is necessarily some nonzero concentration of such defects appearing while the crystal is at the high temperatures seen during growth. The high-temperature defect densities are frozen into the lattice as the crystal cools.

Frenkle defects result when an atom moves away from a lattice site and assumes a position between lattice sites (i.e., takes an interstitial position), as shown in Fig. 1.9(b). The Frenkle defect, therefore, corresponds to a pair of defects, namely, the empty lattice site and the extra interstitially positioned atom. The activation

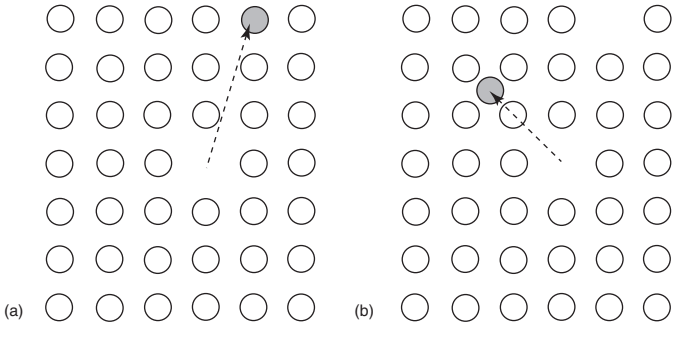


FIGURE 1.8 Point defects in semiconductors: (a) Schottky defects, (b) Frenkle defects.

© 2006 by Taylor & Francis Group, LLC

energy E_{fd} required for formation of this defect pair again establishes a nonzero equilibrium concentration N_{fd} of Frenkle defects given by $N_{fd} = \sqrt{N_L N_I} \exp(-E_{fd}/kT)$. Again, the Frenkle defects tend to form during crystal growth and are frozen into the lattice as the crystal crystallizes.

These point defects significantly impact semiconductor crystals formed mainly through ionic bonding. The covalent bonds of group IV and the largely covalent bonds of group III–V semiconductors, however, are much stronger than ionic bonds, leading to much higher activation energies for point defects and a correspondingly lower defect density in semiconductors with fully or largely covalent bonds. For this reason, the Schottky and Frenkle defects are not of primary importance in the electronic behavior of typical IV–IV and III–V semiconductors.

Line Defects

Three major types of line defects (*edge dislocations, screw dislocations*, and *antiphase defects*) are summarized here. The line defects are generally important factors in the electrical behavior of semiconductor devices since their influence (as trapping centers, scattering sites, etc.) extends over distances substantially larger than atomic spacings.

Edge dislocations correspond to an extra plane inserted orthogonal to the growth direction of a crystal, as illustrated in Fig. 1.9(a). The crystalline lattice is disrupted in the region near where the extra plane starts, leaving behind a line of *dangling bonds*, as shown. The dangling bonds can lead to a variety of effects. The atoms at the start of the dislocation are missing a shared electron in their outer band, suggesting that they may act as traps, producing a linear chain of trap sites with interatomic spacings (vastly smaller than normally encountered between intentionally introduced acceptor impurities. In addition to their impact on the electrical performance, such defects can compromise the mechanical strength of the crystal.

Screw dislocations result from an extra plane of atoms appearing at the surface. In this case (Fig. 1.9(b)), the growth process leads to a spiral structure growing vertically (from the edge of the extra plane). The change in lattice structure over an extended region as the crystal grows can introduce a variety of changes in the etching properties of the crystal, degraded junctions, etc. In addition, the dangling bonds act as traps, impacting electrical characteristics.

Antiphase defects occur in compound semiconductors. Figure 1.9(c) illustrates a section of a III–V semiconductor crystal in which the layers of alternating Ga and As atoms on the right side is one layer out of phase relative to the layers on the left side. This phase error leads to bonding defects in the plane joining the two sides, impacting the electrical performance of devices. Such defects require precision engineering of the growth of large diameter crystals to ensure that entire planes form simultaneously.

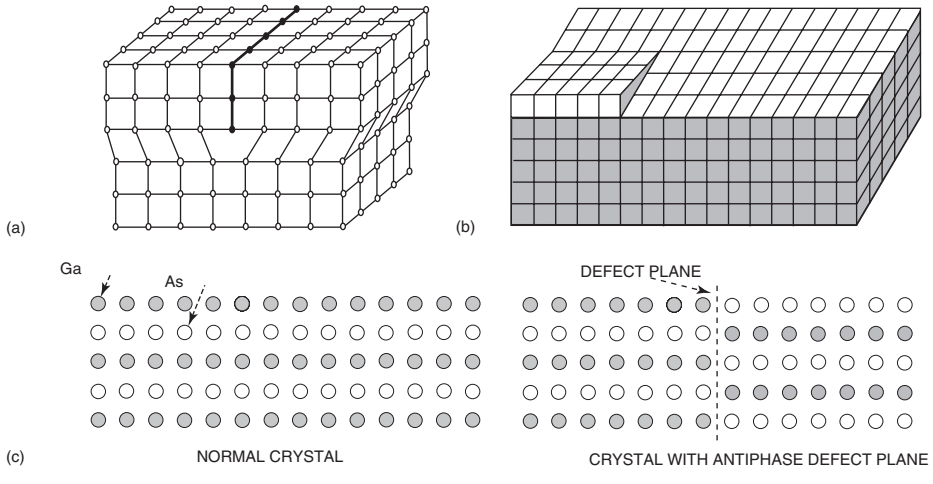


FIGURE 1.9 Line defects: (a) edge dislocation, (b) screw dislocation, (c) antiphase defect.

© 2006 by Taylor & Francis Group, LLC

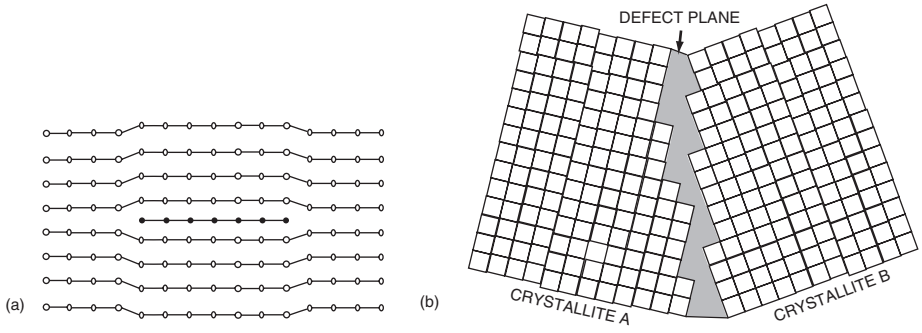


FIGURE 1.10 (a) Stacking faults, (b) grain boundary defects.

Stacking Faults and Grain Boundaries

Stacking faults result when an extra, small area plane (platelet) of atoms occurs within the larger crystal. The result is that normal planes are distorted to extend over that platelet (Fig. 1.10(a)). Here, a stack of defects appears above the location of the platelet. Electrical properties are impacted due to traps and other effects.

Grain boundaries appear when two growing microcrystals with different crystallographic orientations merge, leaving a line of defects along the plane separating the two crystalline lattices as shown in Fig. 1.10(b). As illustrated, the region between the two misoriented crystal lattices is filled from each of the lattices, leaving a plane of defects. Grain boundaries present barriers to conduction at the boundary and produce traps and other electrical effects. In the case of polysilicon (small grains of silicon crystal in the polysilicon area), the behavior includes the effect of the silicon crystal seen within the grains as well as the effects of grain boundary regions (i.e., acts as an interconnected network of crystallites).

Unintentional Impurities

Chemicals and environments encountered during crystal growth and during device microfabrication contain small amounts of impurities that can be incorporated into the semiconductor crystal lattice. Some unintentional impurities replace atoms at lattice sites and are called **substitutional impurities.** Others take positions between lattice sites and are called **interstitial impurities**. Some impurities are benign in the sense of not strongly impacting the behavior of devices. Others are beneficial, improving device performance. Examples include hydrogen, which can compensate dangling bonds and elements that give rise to deep energy trapping levels (i.e., energy levels near the center of the band gap). Such deep level trapping levels are important in indirect gap semiconductors in establishing recombination time constants. In fact, gold was deliberately incorporated in earlier silicon bipolar transistors to increase the recombination rate and achieve faster transistors. Finally, other impurities are detrimental to the device performance.

Optoelectronic devices are often more sensitive to unintentionally occurring impurities. A variety of characteristic trap levels are associated with several of the defects encountered. Table 1.12 summarizes several established trap levels in GaAs, some caused by unintentional impurities and others caused by defects. The strategy for semiconductor material growth and microfabrication is, therefore, a complex strategy, carefully minimizing those unintentional impurities (e.g., sodium in the gate oxides of silicon metal oxide semiconductor (MOS) transistors), selectively adding impurities at appropriate levels to advantage, and maintaining a reasonable strategy regarding unintentional impurities which are benign.

Surface Defects: The Reconstructed Surface

If one imagines slicing a single crystal along a crystallographic plane, a very high density of atomic bonds is broken between the two halves. Such dangling bonds would represent the surface of the crystal, if

© 2006 by Taylor & Francis Group, LLC

TABLE 1.12 Trap States (with Energy E_t) in GaAs

Type	$E_c - E_t$	Name	Type	$E_t - E_v$	Name
Shallow donor	≈5.8 meV		Shallow acceptor	≈10 meV	
Oxygen donor	0.82 eV	EL2	Tin acceptor	0.17 eV	
Chromium acceptor	0.61 eV	EL1	Copper acceptor	0.42 eV	
Deep acceptor	0.58 eV	EL3	Hole trap	0.71 eV	HB2
Electron trap	0.90 eV	EB3	Hole trap	0.29 eV	HB6
Electron trap	0.41 eV	EB6	Hole trap	0.15 eV	HC1

Source: Adapted from Shur, M. 1987. *GaAs Devices and Circuits.* Plenum Press, New York.

the crystal structure extended fully to the surface. The atomic lattice and broken bonds implied by the slicing, however, do not represent a minimum energy state for the surface. As a result, there is a reordering of bonds and atomic sites at the surface to achieve a minimum energy structure. The process is called *surface reconstruction* and leads to a substantially different lattice structure at the surface. This surface reconstruction will be highly sensitive to a variety of conditions, making the surface structure in real semiconductor crystals quite complex. Particularly important are any surface layers (e.g., the native oxide on Si semiconductors) that can incorporate atoms different from the semiconductor's basis atoms. The importance of surfaces is clearly seen in Si MOS transistors, where surface interfaces with low defect densities are now readily achieved. The reconstructed surface can significantly impact electrical properties of several devices. For example, mobilities of carriers moving in the surface region can be substantially reduced compared to bulk mobilities. In addition, MOS devices are sensitive to fixed charge and trap sites that can moderate the voltage appearing at the semiconductor surface relative to the applied gate voltage.

Surface reconstruction is a very complex and detailed topic and is not considered further here. Typically, microfabrication techniques have carefully evolved to achieve high-quality surfaces.

1.6 Summary

A rich diversity of semiconductor materials, led by the extraordinarily advanced device technologies for silicon microelectronics and III–V optoelectronics, has been explored for a wide range of applications. Much of the computing, information, and consumer electronics revolutions expected over the next decade will rest of the foundation of these important crystalline materials. As established semiconductor materials such as silicon continue to define the frontier for advanced fabrication of very small devices, new materials are emerging or being reconsidered as possible additional capabilities for higher speed, lower power, and other advantages. This chapter has provided a broad overview of semiconductor materials. Many fine and highly readable books, listed in the references, provide the additional depth which could not be provided in this brief chapter. Tables of various properties are provided throughout the chapter, with much of the relevant information provided in Table 1.13.

Defining Terms

Ambipolar semiconductors: Semiconductors that can be selectively doped to achieve either *n* type or *p* type material.

Amphoteric impurities: Doping impurities that may act as either a donor or an acceptor.

Bandgap: Energy difference between the conduction band and the valence band.

Compound semiconductors: Semiconductor crystals composed of two or more atomic elements from different groups of the periodic chart.

Conduction band: Energy level for free electrons at rest in the semiconductor crystal.

Deep energy level impurities: Doping impurities or other impurities whose energy level lies toward the center of the bandgap. Important for carrier recombination in indirect gap semiconductors.

© 2006 by Taylor & Francis Group, LLC

TABLE 1.13 Properties of GaAs and Si Semiconductors at Room Temperature (300 K), CdS and CdSe Both Can Appear in Either Zinc Blende or Wurtzite Lattice Forms

	Lattice	Lattice Constant, Å	Energy Gap, eV	Electron Effective Mass	Hole Effective Mass	Dielectric Constant	Electron Mobility, cm^2/V-s	Hole Mobility, cm^2/V-s
C	Diamond	3.5668	5.47 (I)[a]	0.2	0.25	5.7	1,800	1,200
Si	Diamond	5.4310	1.11 (I)	m_l: 0.98 m_t: 0.19	m_l: 0.16 m_h: 0.5	11.7	1,350	480
Ge	Diamond	5.6461	0.67 (I)	m_l: 1.58 m_t: 0.08	m_l: 0.04 m_h: 0.3	16.3	3,900	1,900
AlP	Zinc blende	5.4625	2.43	0.13		9.8	80	
AlAs	Zinc blende	5.6605	2.16 (I)	0.5	m_l: 0.49 m_h: 1.06	12.0	1,000	180
AlSb	Zinc blende	6.1355	1.52 (I)	0.11	m_l: 0.39	11	900	400
GaN	Wurtzite	3.189	3.4	0.2	0.8	12	300	
GaP	Zinc blende	5.4506	2.26 (I)	0.13	0.67	10	300	150
GaAs	Zinc blende	5.6535	1.43 (D)[b]	0.067	m_l: 0.12 m_h: 0.5	12.5	8,500	400
GaSb	Zinc blende	6.0954	0.72 (D)	0.045	0.39	15.0	5,000	1,000
InAs	Zinc blende	6.0584	0.36 (D)	0.028	0.33	12.5	22,600	200
InSb	Zinc blende	6.4788	0.18 (D)	0.013	0.18	18	100,000	1,700
InP	Zinc blende	5.8687	1.35 (D)	0.077	m_l: 0.12 m_h: 0.60	12.1	4,000	600
CdS	Zinc blende	5.83	2.42 (D)	0.2	0.7	5.4	340	50
CdSe	Zinc blende	6.05	1.73 (D)	0.13	0.4	10.0	800	
CdTe	Zinc blende	6.4816	1.50 (D)	0.11	0.35	10.2	1,050	100
PbS	NaCl	5.936	0.37 (I)	0.1	0.1	17.0	500	600
PbSe	NaCl	6.147	0.26 (I)	m_l: 0.07 m_t: 0.039	m_l: 0.06 m_h: 0.03	23.6	1,800	930
PbTe	NaCl	6.45	0.29 (I)	m_l: 0.24 m_t: 0.02	m_l: 0.3 m_h: 0.02	30	6,000	4,100

[a]Indirect bands.

[b]Direct bands.

Source: Adapted from Wolfe, C.M., Holonyak, N., and Stillman, G.E. 1989. *Physical Properties of Semiconductors*. Prentice-Hall, New York.

Direct gap semiconductor: Semiconductor whose conduction band minimum and valance band minimum appear at the same wave vector (same momenta); important for optical sources.

Effective mass: Value of carrier mass used to represent motion of a carrier in the semiconductor as though the motion were in free space.

Elemental semiconductors: Semiconductor crystals composed of a single atomic element.

Indirect gap semiconductor: Semiconductor whose conduction band minimum and valence band minimum appear at different wave vectors (different momenta).

Intrinsic semiconductors: Semiconductors with no intentional or nonintentional impurities (dopants).

Low field mobility: Proportionality constant between carrier velocity and electric field.

Quaternary semiconductors: Compound semiconductors with two atomic elements from one group and two atomic elements from a second group of the periodic chart.

Saturated velocities: Maximum carrier velocity induced by electric field (due to onset of inelastic scattering).

Shallow energy level dopants: Doping impurities whose energy level lies very close to the conduction (valence) band for donors (acceptors).

Substitutional impurities: Impurities that replace the crystal's base atom at that base atom's lattice position.

© 2006 by Taylor & Francis Group, LLC

Ternary semiconductors: Compound semiconductors with two atomic elements from one group and one atomic element from a second group of the periodic chart.

Unipolar semiconductors: Semiconductors that can be selectively doped to achieve only *n* type or only *p* type material.

Valence band: Energy for bound electrons and for free holes at rest in the semiconductor crystal.

References

Beadle, W.E., Tsai, J.C.C., and Plummer, R.D., eds., 1985. *Quick Reference Manual for Silicon Integrated Circuit Technology.* Wiley, New York.

Böer, K.W. 1990. *Survey of Semiconductor Physics, Vol. 1: Electrons and Other Particles in Bulk Semiconductors.* Van Nostrand, New York.

Böer, K.W. 1992. *Survey of Semiconductor Physics, Vol. 2: Barriers, Junctions, Surfaces, and Devices.* Van Nostrand, New York.

Capasso, F. and Margaritondo, G., eds. 1987. *Heterojunction and Band Discontinuities.* North Holland, Amsterdam.

Haug, A. 1975. *Theoretical Solid State Physics, Vols. 1 and 2*. Pergamon Press, Oxford, England.

Howes, M.J. and Morgan, D.V. 1985. *Gallium Arsenide: Materials, Devices, and Circuits.* Wiley, New York.

Irvine, S.J.C., Lum, B., Mullin, J.B., and Woods, J., eds. 1982. *II–VI Compounds 1982.* North Holland, Amsterdam.

Lannoo, M. and Bourgoin, J. 1981. *Point Defects in Semiconductors.* Springer-Verlag, Berlin.

Loewrro, M.H., ed. 1985. *Dislocations and Properties of Real Materials.* Inst. of Metals, London.

Moss, T.S. and Balkanski, M., eds. 1980. *Handbook on Semiconductors, Vol. 2: Optical Properties of Solids.* North Holland, Amsterdam.

Moss, T.S. and Keller, S.P., eds. 1980. *Handbook on Semiconductors, Vol. 3: Material Properties and Preparation.* North Holland, Amsterdam.

Moss, T.S. and Paul, W. 1982. *Handbook on Semiconductors, Vol. 1: Band Theory and Transport Properties.* North Holland, Amsterdam.

Nicollian, E.H. and Brews, J.R. 1982. *MOS Physics and Technology.* Wiley, New York.

Pantelides, S.T. 1986. *Deep Centers in Semiconductors.* Gordon & Breach Science, New York.

Shur, M. 1987. *GaAs Devices and Circuits.* Plenum Press, New York.

Smith, R.A. 1978. *Semiconductors.* Cambridge University Press, London.

Sze, S.M. 1981. *Physics of Semiconductor Devices*, 2nd ed. Wiley Interscience, New York.

Tyagi, M.S. 1991. *Introduction to Semiconductor Materials.* Wiley, New York.

Wang, S. 1989. *Fundamentals of Semiconductor Theory and Device Physics.* Prentice-Hall, Englewood Cliffs, New Jersey.

Willardson, A.K. and Beer, A.C., eds. 1985. *Semiconductors and Semimetals, Vol. 22.* Academic Press, New York.

Wilmsen, C.W., ed. 1985a. *Physics and Chemistry of III–V Compounds.* Plenum Press, New York.

Wilmsen, C.W., ed. 1985b. *Physics and Chemistry of III–V Compound Semiconductor Interfaces.* Plenum Press, New York.

Wolfe, C.M., Holonyak, N., and Stillman, G.E. 1989. *Physical Properties of Semiconductors.* Prentice-Hall, New York.

Further Information

The list of references focuses on books that have become popular in the field. The reader is advised to refer to such books for additional detail on semiconductor materials. The Institute for Electrical and Electronics Engineers (IEEE, Piscataway, NJ) publishes several journals that provide information on recent advances in materials and their uses in devices. Examples include the *IEEE Trans. Electron Devices*, the *IEEE Solid State Circuits Journal*, the *IEEE Journal on Quantum Electronics*, and the *IEEE Journal on*

© 2006 by Taylor & Francis Group, LLC

Lightwave Technology. The American Physical Society also publishes several journals focusing on semiconductor materials. One providing continuing studies of materials is the *Journal of Applied Physics*. The Optical Society of America (OSA, Washington, DC) publishes *Applied Optics*, a journal covering many practical issues as well as new directions. The Society of Photo-Optical Instrumentation Engineers (SPIE, Bellingham, WA) sponsors a broad range of conferences, with an emphasis on optoelectronic materials but also covering other materials. The SPIE can be contacted directly for a list of its extensive set of conference proceedings.

© 2006 by Taylor & Francis Group, LLC

2

Thermal Properties

2.1 Introduction 2-1

2.2 Fundamentals of Heat 2-1

Temperature • Heat Capacity • Specific Heat • Thermal Conductivity • Thermal Expansion • Solids • Liquids • Gases

2.3 Other Material Properties 2-4

Insulators • Dielectric Constant • Resistivity • Semiconductors • Conductors • Melting Point

2.4 Engineering Data 2-5

Temperature Coefficient of Capacitance • Temperature Coefficient of Resistance • Temperature Compensation

David F. Besch

2.1 Introduction

The rating of an electronic or electrical device depends on the capability of the device to dissipate heat. As miniaturization continues, engineers are more concerned about heat dissipation and the change in properties of the device and its material makeup with respect to temperature.

The following section focuses on heat and its result. Materials may be categorized in a number of different ways. In this chapter, materials will be organized in the general classifications according to their resistivities:

- Insulators
- Semiconductors
- Conductors

It is understood that with this breakdown, some materials will fit naturally into more than one category. Ceramics, for example, are insulators, yet with alloying of various other elements, can be classified as semiconductors, resistors, a form of conductor, and even conductors. Although, in general, the change in resistivity with respect to temperature of a material is of interest to all, the design engineer is more concerned with how much a resistor changes with temperature and if the change drives the circuit parameters out of specification.

2.2 Fundamentals of Heat

In the commonly used model for materials, heat is a form of energy associated with the position and motion of the material's molecules, atoms, and ions. The position is analogous with the state of the material and is potential energy, whereas the motion of the molecules, atoms, and ions is kinetic energy. Heat added to a material makes it hotter and vice versa. Heat also can melt a solid into a liquid and convert liquids into gases, both changes of state. Heat energy is measured in calories (cal), British thermal units (Btu), or joules (J). A calorie is the amount of energy required to raise the temperature of one gram (1 g) of water one degree Celsius (1°C) (14.5 to 15.5°C). A Btu is a unit of energy necessary to raise the temperature of

© 2006 by Taylor & Francis Group, LLC

one pound (1 lb) of water by one degree Fahrenheit (1°F). A joule is an equivalent amount of energy equal to work done when a force of one newton (1 N) acts through a distance of one meter (1 m). Thus heat energy can be turned into mechanical energy to do work. The relationship among the three measures is: 1 Btu = 251.996 cal = 1054.8 J.

Temperature

Temperature is a measure of the average kinetic energy of a substance. It can also be considered a relative measure of the difference of the heat content between bodies. Temperature is measured on either the Fahrenheit scale or the Celsius scale. The Fahrenheit scale registers the freezing point of water as 32°F and the boiling point as 212°F. The Celsius scale or centigrade scale (old) registers the freezing point of water as 0°C and the boiling point as 100°C.

The Rankine scale is an absolute temperature scale based on the Fahrenheit scale. The Kevin scale is an absolute temperature scale based on the Celsius scale. The absolute scales are those in which zero degree corresponds with zero pressure on the hydrogen thermometer. For the definition of temperature just given, zero °R and zero K register zero kinetic energy.

The four scales are related by the following:

$$^\circ\mathrm{C} = 5/9(^\circ\mathrm{F} - 32)$$
$$^\circ\mathrm{F} = 9/5(^\circ\mathrm{C}) + 32$$
$$\mathrm{K} = ^\circ\mathrm{C} + 273.16$$
$$^\circ\mathrm{R} = ^\circ\mathrm{F} + 459.69$$

Heat Capacity

Heat capacity is defined as the amount of heat energy required to raise the temperature of one mole or atom of a material by 1°C without changing the state of the material. Thus it is the ratio of the change in heat energy of a unit mass of a substance to its change in temperature. The heat capacity, often called thermal capacity, is a characteristic of a material and is measured in cal/g per °C or Btu/lb per °F,

$$c_p = \frac{\partial H}{\partial T}$$

Specific Heat

Specific heat is the ratio of the heat capacity of a material to the heat capacity of a reference material, usually water. Since the heat capacity of water is 1 Btu/lb and 1 cal/g, the specific heat is numerically equal to the heat capacity.

Thermal Conductivity

Heat transfers through a material by conduction resulting when the energy of atomic and molecular vibrations is passed to atoms and molecules with lower energy. In addition, energy flows due to free electrons,

$$Q = kA\frac{\partial T}{\partial l}$$

where

Q = heat flow per unit time
k = thermal conductivity
A = area of thermal path
l = length of thermal path
T = temperature

The coefficient of thermal conductivity k is temperature sensitive and decreases as the temperature is raised above room temperature.

© 2006 by Taylor & Francis Group, LLC

Thermal Expansion

As heat is added to a substance the kinetic energy of the lattice atoms and molecules increases. This, in turn, causes an expansion of the material that is proportional to the temperature change, over normal temperature ranges. If a material is restrained from expanding or contracting during heating and cooling, internal stress is established in the material.

$$\frac{\partial l}{\partial T} = \beta_L l \qquad \text{and} \qquad \frac{\partial V}{\partial T} = \beta_V V$$

where

l = length
V = volume
T = temperature
β_L = coefficient of linear expansion
β_V = coefficient of volume expansion

Solids

Solids are materials in a state in which the energy of attraction between atoms or molecules is greater than the kinetic energy of the vibrating atoms or molecules. This atomic attraction causes most materials to form into a crystal structure. Noncrystalline solids are called *amorphous*, including glasses, a majority of plastics, and some metals in a semistable state resulting from being cooled rapidly from the liquid state. Amorphous materials lack a long range order.

Crystalline materials will solidify into one of the following geometric patterns:

- Cubic
- Tetragonal
- Orthorhombic
- Monoclinic
- Triclinic
- Hexagonal
- Rhombohedral

Often the properties of a material will be a function of the density and direction of the lattice plane of the crystal.

Some materials will undergo a change of state while still solid. As it is heated, pure iron changes from body centered cubic to face centered cubic at 912°C with a corresponding increase in atomic radius from 0.12 to 0.129 nm due to thermal expansion. Materials that can have two or more distinct types of crystals with the same composition are called *polymorphic.*

Liquids

Liquids are materials in a state in which the energies of the atomic or molecular vibrations are approximately equal to the energy of their attraction. Liquids flow under their own mass. The change from solid to liquid is called *melting*. Materials need a characteristic amount of heat to be melted, called the *heat of fusion.* During melting the atomic crystal experiences a disorder that increases the volume of most materials. A few materials, like water, with **stereospecific** covalent bonds and low packing factors attain a denser structure when they are thermally excited.

Gases

Gases are materials in a state in which the kinetic energies of the atomic and molecular oscillations are much greater than the energy of attraction. For a given pressure, gas expands in proportion to the absolute

© 2006 by Taylor & Francis Group, LLC

temperature. For a given volume, the absolute pressure of a gas varies in proportion to the absolute pressure. For a given temperature, the volume of a given weight of gas varies inversely as the absolute pressure. These three facts can be summed up into the Gas law:

$$PV = RT$$

where

P = absolute pressure
V = specific volume
T = absolute temperature
R = universal gas constant t

Materials need a characteristic amount of heat to transform from liquid to solid, called the *heat of vaporization.*

2.3 Other Material Properties

Insulators

Insulators are materials with resistivities greater than about $10^7\ \Omega \cdot$ cm. Most ceramics, plastics, various oxides, paper, and air are all insulators. Alumina (Al_2O_3) and beryllia (BeO) are ceramics used as substrates and chip carriers. Some ceramics and plastic films serve as the dielectric for capacitors.

Dielectric Constant

A capacitor consists of two conductive plates separated by a dielectric. Capacitance is directly proportional to the dielectric constant of the insulating material. Ceramic compounds doped with barium titanate have high dielectric constants and are used in capacitors. Plastics, such as mica, polystyrene, polycarbonate, and polyester films also serve as dielectrics for capacitors. Capacitor values are available with both positive and negative changes in value with increased temperature. See the first subsection in Sec. 2.4 for a method to calculate the change in capacitor values at different temperatures.

Resistivity

The resistivity of insulators typically decreases with increasing temperature.

Semiconductors

Semiconductors are materials that range in resistivity from approximately 10^{-4} to $10^{+7}\ \Omega\cdot$ cm. Silicon (Si), Germanium (Ge), and Gallium Arsenide (GaAs) are typical semiconductors. The resistivity and its inverse, the conductivity, vary over a wide range, due primarily to doping of other elements. The conductivity of intrinsic Si and Ge follows an exponential function of temperature,

$$\sigma = \sigma_0 e^{\frac{E_g}{2kT}}$$

where

σ = conductivity
σ_0 = constant t
E_g = 1.1 eV for Si
k = Bolzmann's constant t
T = temperature °K

Thus, the electrical conductivity of Si increases by a factor of 2400 when the temperatures rises from 27 to 200 K.

© 2006 by Taylor & Francis Group, LLC

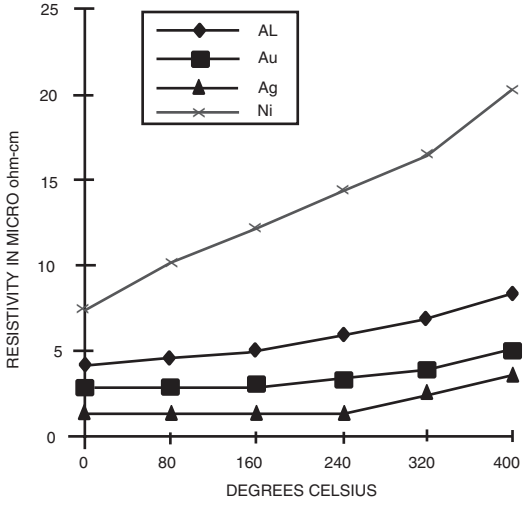


FIGURE 2.1 Resistivity in a function of temperature.

Conductors

Conductors have resistivity value less than 10^{-4} Ω· cm and include metals, metal oxides, and conductive nonmetals. The resistivity of conductors typically increases with increasing temperature as shown in Fig. 2.1

Melting Point

Solder is an important material used in electronic systems. The tin-lead solder system is the most used solder compositions. The system's equilibrium diagram shows a typical eutectic at 61.9% Sn. Alloys around the eutectic are useful for general soldering. High Pb content solders have up to 10% Sn and are useful as high-temperature solders. High Sn solders are used in special cases such as in high corrosive environments. Some useful alloys are listed in Table 2.1.

2.4 Engineering Data

Graphs of resistivity and dielectric constant vs. temperature are difficult to translate to values of electronic components. The electronic design engineer is more concerned with how much a resistor changes with temperature and if the change drives the circuit parameters out of specification. The following defines the commonly used terms for components related to temperature variation.

TABLE 2.1 Characteristics of certain alloys

Sn (%)	Pb (%)	Ag (%)	°C
60	40	—	190
60	38	2	192
10	90	—	302
90	10	—	213
95	5	5	230

© 2006 by Taylor & Francis Group, LLC

Temperature Coefficient of Capacitance

Capacitor values vary with temperature due to the change in the dielectric constant with temperature change. The temperature coefficient of capacitance (TCC) is expressed as this change in capacitance with a change in temperature.

$$\text{TCC} = \frac{1}{C}\frac{\partial C}{\partial T}$$

where

TCC = temperature coefficient of capacitance
C = capacitor value
T = temperature

The TCC is usually expressed in parts per million per degree Celsius (ppm/°C). Values of TCC may be positive, negative, or zero. If the TCC is positive, the capacitor will be marked with a P preceding the numerical value of the TCC. If negative, N will precede the value. Capacitors are marked with NPO if there is no change in value with a change in temperature. For example, a capacitor marked N1500 has a $-$ 1500/1,000,000 change in value per each degree Celsius change in temperature.

Temperature Coefficient of Resistance

Resistors change in value due to the variation in resistivity with temperature change. The temperature coefficient of resistance (TCR) represents this change. The TCR is usually expressed in parts per million per degree Celsius (ppm/°C).

$$\text{TCR} = \frac{1}{R}\frac{\partial R}{\partial T}$$

where

TCR = temperature coefficient of resisitance
R = resistance value
T = temperature

Values of TCR may be positive, negative, or zero. TCR values for often used resistors are shown in Table 2.2. The last three TCR values refer to resistors imbedded in silicon monolithic integrated circuits.

Temperature Compensation

Temperature compensation refers to the active attempt by the design engineer to improve the performance and stability of an electronic circuit or system by minimizing the effects of temperature change. In addition to utilizing optimum TCC and TCR values of capacitors and resistors, the following components and techniques can also be explored.

TABLE 2.2 TCR values of common resistors

Resistor Type	TCR, ppm/°C		
Carbon composition	+500	to	+2000
Wire wound	+200	to	+500
Thick film	+20	to	+200
Thin film	+20	to	+100
Base diffused	+1500	to	+2000
Emitter diffused	+600		
Ion implanted	±100		

© 2006 by Taylor & Francis Group, LLC

- Thermistors
- Circuit design stability analysis
- Thermal analysis

Thermistors

Thermistors are semiconductor resistors that have resistor values that vary over a wide range. They are available with both positive and negative temperature coefficients and are used for temperature measurements and control systems, as well as for temperature compensation. In the latter they are utilized to offset unwanted increases or decreases in resistance due to temperature change.

Circuit Analysis

Analog circuits with semiconductor devices have potential problems with bias stability due to changes in temperature. The current through junction devices is an exponential function as follows:

$$i_D = I_S\left(e^{\frac{qv_D}{nkT}} - 1\right)$$

where

i_D = junction current
I_S = saturation current
v_D = junction voltage
q = electron charge
n = emission coefficient
k = Boltzmann's constant
T = temperature, in 0 K

Junction diodes and bipolar junction transistor currents have this exponential form. Some biasing circuits have better temperature stability than others. The designer can evaluate a circuit by finding its fractional temperature coefficient,

$$\mathrm{TC}_F = \frac{1}{v(T)}\frac{\partial v(T)}{\partial T}$$

where

$v(T)$ = circuit variable
TC_F = temperature coefficient
T = temperature

Commercially available circuit simulation programs are useful for evaluating a given circuit for the result of temperature change. SPICE, for example, will run simulations at any temperature with elaborate models included for all circuit components.

Thermal Analysis

Electronic systems that are small or that dissipate high power are subject to increases in internal temperature. Thermal analysis is a technique in which the designer evaluates the heat transfer from active devices that dissipate power to the ambient.

Defining Terms

Eutectic: Alloy composition with minimum melting temperature at the intersection of two solubility curves.

Stereospecific: Directional covalent bonding between two atoms.

© 2006 by Taylor & Francis Group, LLC

References

Guy, A.G. 1967. *Elements of Physical Metallurgy*, 2nd ed., pp. 255–276. Addison-Wesley, Reading, MA.
Incropera, F.P. and Dewitt, D.P. 1990. *Fundamentals of Heat and Mass Transfer*, 3rd ed., pp. 44–66. Wiley, New York.

Further Information

Additional information on the topic of thermal properties of materials is available from the following sources:

Banzhaf, W. 1990. *Computer-Aided Circuit Analysis Using Psice*. Prentice-Hall, Englewood Cliffs, NJ.
Smith, W.F. 1990. *Principles of Material Science and Engineering*. McGraw-Hill, New York.
Van Vlack, L.H. 1980. *Elements of Material Science and Engineering*. Addison-Wesley, Reading, MA.

© 2006 by Taylor & Francis Group, LLC

3

Semiconductors

3.1 Introduction 3-1
Semiconductors
3.2 Diodes 3-2

Sidney Soclof

3.1 Introduction

Transistors form the basis of all modern electronic devices and systems, including the integrated circuits used in systems ranging from radio and television to computers. Transistors are solid-state electron devices made out of a category of materials called semiconductors. The mostly widely used semiconductor for transistors, by far, is silicon, although gallium arsenide, which is a compound semiconductor, is used for some very high-speed applications.

Semiconductors

Semiconductors are a category of materials with an electrical conductivity that is between that of conductors and insulators. Good conductors, which are all metals, have electrical resistivities down in the range of 10^{-6} Ω-cm. Insulators have electrical resistivities that are up in the range from 10^{6} to as much as about 10^{12} Ω-cm. Semiconductors have resistivities that are generally in the range of 10^{-4}–10^{4} Ω-cm. The resistivity of a semiconductor is strongly influenced by impurities, called **dopants**, that are purposely added to the material to change the electronic characteristics.

We will first consider the case of the pure, or **intrinsic semiconductor**. As a result of the thermal energy present in the material, electrons can break loose from covalent bonds and become free electrons able to move through the solid and contribute to the electrical conductivity. The covalent bonds left behind have an electron vacancy called a **hole**. Electrons from neighboring covalent bonds can easily move into an adjacent bond with an electron vacancy, or hole, and thus the hold can move from one covalent bond to an adjacent bond. As this process continues, we can say that the hole is moving through the material. These holes act as if they have a positive charge equal in magnitude to the electron charge, and they can also contribute to the electrical conductivity. Thus, in a semiconductor there are two types of mobile electrical charge carriers that can contribute to the electrical conductivity, the free electrons and the holes. Since the electrons and holes are generated in equal numbers, and recombine in equal numbers, the free electron and hole populations are equal.

In the **extrinsic** or **doped semiconductor**, impurities are purposely added to modify the electronic characteristics. In the case of silicon, every silicon atom shares its four valence electrons with each of its four nearest neighbors in covalent bonds. If an impurity or dopant atom with a valency of five, such as phosphorus, is substituted for silicon, four of the five valence electrons of the dopant atom will be held in covalent bonds. The extra, or fifth electron will not be in a covalent bond, and is loosely held. At room temperature, almost all of these extra electrons will have broken loose from their parent atoms, and become free electrons. These pentavalent dopants thus donate free electrons to the semiconductor and are called **donors**. These donated electrons upset the balance between the electron and hole populations, so there are

© 2006 by Taylor & Francis Group, LLC

now more electrons than holes. This is now called an **N-type semiconductor**, in which the electrons are the **majority carriers**, and holes are the **minority carriers**. In an N-type semiconductor the free electron concentration is generally many orders of magnitude larger than the hole concentration.

If an impurity or dopant atom with a valency of three, such as boron, is substituted for silicon, three of the four valence electrons of the dopant atom will be held in covalent bonds. One of the covalent bonds will be missing an electron. An electron from a neighboring silicon-to-silicon covalent bond, however, can easily jump into this electron vacancy, thereby creating a vacancy, or hole, in the silicon-to-silicon covalent bond. Thus, these trivalent dopants accept free electrons, thereby generating holes, and are called **acceptors**. These additional holes upset the balance between the electron and hole populations, and so there are now more holes than electrons. This is called a **P-type semiconductor**, in which the holes are the majority carriers, and the electrons are the minority carriers. In a P-type semiconductor the hole concentration is generally many orders of magnitude larger than the electron concentration.

Figure 3.1 shows a single crystal chip of silicon that is doped with acceptors to make it P-type on one side, and doped with donors to make it N-type on the other side. The transition between the two sides is called the *PN junction*. As a result of the concentration difference of the free electrons and holes there will be an initial flow of these charge carriers across the junction, which will result in the N-type side attaining a net positive charge with respect to the P-type side. This results in the formation of an electric potential *hill* or barrier at the junction. Under equilibrium conditions the height of this potential hill, called the **contact potential** is such that the flow of the majority carrier holes from the P-type side up the hill to the N-type side is reduced to the extent that it becomes equal to the flow of the minority carrier holes from the N-type side down the hill to the P-type side. Similarly, the flow of the majority carrier free electrons from the N-type side is reduced to the extent that it becomes equal to the flow of the minority carrier electrons from the P-type side. Thus, the net current flow across the junction under equilibrium conditions is zero.

FIGURE 3.1 PN junction.

3.2 Diodes

In Fig. 3.2 the silicon chip is connected as a *diode* or two-terminal electron device. The situation in which a bias voltage is applied is shown. In Fig. 3.2(a) the bias voltage is a **forward bias**, which produces a reduction in the height of the potential hill at the junction. This allows for a large increase in the flow of electrons and holes across the junction. As the forward bias voltage increases, the **forward current** will increase at approximately an exponential rate, and can become very large. The variation of forward current flow with forward bias voltage is given by the diode equation as

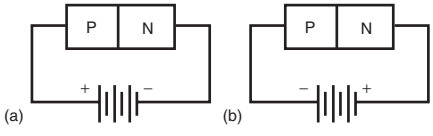


FIGURE 3.2 Biasing of a diode: (a) forward bias, (b) reverse bias.

$$I = I_0(\exp(qV/nkT) - 1)$$

where

I_0 = reverse saturation current, constant
q = electron charge
n = dimensionless factor between 1 and 2
k = Boltzmann's constant
T = absolute temperature, K

If we define the **thermal voltage** as $V_T = kT/q$, the diode equation can be written as

$$I = I_0(\exp(V/nV_T) - 1)$$

At room temperature $V_T \cong 26\ mV$, and n is typically around 1.5 for silicon diodes.

© 2006 by Taylor & Francis Group, LLC

In Fig. 3.2(b) the bias voltage is a **reverse bias**, which produces an increase in the height of the potential hill at the junction. This essentially chokes off the flow of electrons from the N-type side to the P-type side, and holes from the P-type side to the N-type side. The only thing left is the very small trickle of electrons from the P-type side and holes from the N-type side. Thus the **reverse current** of the diode will be very small.

In Fig. 3.3 the circuit schematic symbol for the diode is shown, and in Fig. 3.4 a graph of the current vs. voltage curve for the diode is presented. The P-type side of the diode is called the **anode**, and the N-type side is the **cathode** of the diode. The forward current of diodes can be very large, in the case of large power diodes, up into the range of 10–100 A. The reverse current is generally very small, often down in the low nanoampere, or even picoampere range. The diode is basically a one-way voltage-controlled current switch. It allows current to flow in the forward direction when a forward bias voltage is applied, but when a reverse bias is applied the current flow becomes extremely small. Diodes are used extensively in electronic circuits. Applications include rectifiers to convert AC to DC, wave shaping circuits, peak detectors, DC level shifting circuits, and signal transmission gates. Diodes are also used for the demodulation of amplitude-modulated (AM) signals.

FIGURE 3.3 Diode symbol.

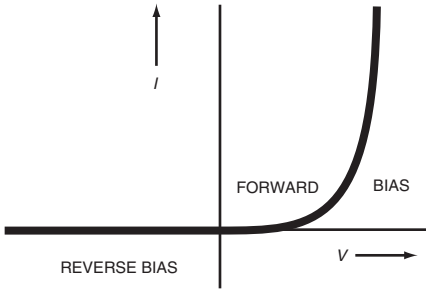


FIGURE 3.4 Current vs. voltage curve for a diode.

Defining Terms

Acceptors: Impurity atoms that when added to a semiconductor contribute holes. In the case of silicon, acceptors are atoms from the third column of the periodic table, such as boron.

Anode: The P-type side of a diode.

Cathode: The N-type side of a diode.

Contact potential: The internal voltage that exists across a PN junction under thermal equilibrium conditions, when no external bias voltage is applied.

Donors: Impurity atoms that when added to a semiconductor contribute free electrons. In the case of silicon, donors are atoms from the fifth column of the periodic table, such as phosphorus, arsenic, and antimony.

Dopants: Impurity atoms that are added to a semiconductor to modify the electrical conduction characteristics.

Doped semiconductor: A semiconductor that has had impurity atoms added to modify the electrical conduction characteristics.

Extrinsic semiconductor: A semiconductor that has been doped with impurities to modify the electrical conduction characteristics.

Forward bias: A bias voltage applied to the PN junction of a diode or transistor that makes the P-type side positive with respect to the N-type side.

Forward current: The large current flow in a diode that results from the application of a forward bias voltage.

Hole: An electron vacancy in a covalent bond between two atoms in a semiconductor. Holes are mobile charge carriers with an effective charge that is opposite to the charge on an electron.

Intrinsic semiconductor: A semiconductor with a degree of purity such that the electrical characteristics are not significantly affected.

© 2006 by Taylor & Francis Group, LLC

Majority carriers: In a semiconductor, the type of charge carrier with the larger population. For example, in an N-type semiconductor, electrons are the majority carriers.

Minority carriers: In a semiconductor, the type of charge carrier with the smaller population. For example, in an N-type semiconductor, holes are the minority carriers.

N-type semiconductor: A semiconductor that has been doped with donor impurities to produce the condition that the population of free electrons is greater than the population of holes.

P-type semiconductor: A semiconductor that has been doped with acceptor impurities to produce the condition that the population of holes is greater than the population of free electrons.

Reverse bias: A bias voltage applied to the PN junction of a diode or transistor that makes the P-type side negative with respect to the N-type side.

Reverse current: The small current flow in a diode that results from the application of a reverse bias voltage.

Thermal voltage: The quantity kT/q where k is Boltzmann's constant, T is absolute temperature, and q is electron charge. The thermal voltage has units of volts, and is a function only of temperature, being approximately 25 mV at room temperature.

References

Comer, D.J. and Comer, D.T., 2002. *Advanced Electronic Circuit Design*. John Wiley & Sons, New York, NY.

Hambley, A.R. 2000. *Electronics*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ.

Jaeger, R.C. and Travis, B. 2004. *Microelectronic Circuit Design with CD-ROM*. McGraw-Hill, New York.

Mauro, R. 1989. *Engineering Electronics*. Prentice-Hall, Englewood Cliffs, NJ.

Millman, J. and Grabel, A. 1987. *Microelectronics*, 2nd ed. McGraw-Hill, New York.

Mitchell, F.H., Jr. and Mitchell, F.H., Sr. 1992. *Introduction to Electronics Design*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ.

Martin, S., Roden, M.S., Carpenter, G.L., and Wieserman, W.R. 2002. *Electronic Design*, Discovery Press, Los Angeles, CA.

Neamen, D. 2001. Electronic Circuit Analysis with CD-ROM. McGraw-Hill, New York, NY.

Sedra, A.S. and Smith, K.C. 2003. *Microelectronics Circuits*, 5th ed. Oxford University Press, Oxford.

Spencer, R. and Mohammed G. 2003. *Introduction to Electronic Circuit Design*. Prentice-Hall, Englewood Cliffs, NJ.

Further Information

An excellent introduction to the physical electronics of devices is found in Ben G. Streetman, *Solid State Electronic Devices*, 4th ed. Prentice-Hall, Englewood Cliffs, NJ, 1995. Another excellent reference on a wide variety of electronic devices is Kwok K. Ng, *Complete Guide to Semiconductor Devices*, 2002. Wiley-IEEE Computer Society Press, New York, NY. A useful reference on circuits and applications is Donald A. Neamen, *Electronic Circuit Analysis and Design*, 2nd ed., 2001, Irwin, Chicago, IL.

© 2006 by Taylor & Francis Group, LLC

4

Metal-Oxide-Semiconductor Field-Effect Transistor

4.1 Introduction 4-1
4.2 Current-Voltage Characteristics 4-3
Strong-Inversion Characteristics • Subthreshold Characteristics
4.3 Important Device Parameters 4-4
Threshold Voltage • Driving Ability and $I_{D,\mathrm{sat}}$ • Transconductance • Output Resistance and Drain Conductance
4.4 Limitations on Miniaturization 4-10
Subthreshold Control • Hot-Electron Effects • Thin Oxides • Dopant-Ion Control • Other Limitations

John R. Brews

4.1 Introduction

The metal-oxide-semiconductor field-effect transistor (MOSFET) is a transistor that uses a control electrode, the **gate**, to capacitively modulate the conductance of a surface **channel** joining two end contacts, the **source** and the **drain**. The gate is separated from the semiconductor **body** underlying the gate by a thin *gate insulator,* usually silicon dioxide. The surface channel is formed at the interface between the semiconductor body and the gate insulator, see Fig. 4.1.

The MOSFET can be understood by contrast with other field-effect devices, like the junction field-effect transistor (JFET) and the metal-semiconductor field-effect transistor (MESFET) (Hollis and Murphy, 1990). These other transistors modulate the conductance of a *majority-carrier* path between two *ohmic* contacts by capacitive control of its cross section. (Majority carriers are those in greatest abundance in field-free semiconductor, electrons in *n*-type material and holes in *p*-type material.) This modulation of the cross section can take place at any point along the length of the channel, and so the gate electrode can be positioned anywhere and need not extend the entire length of the channel.

Analogous to these field-effect devices is the *buried-channel, depletion-mode,* or *normally on* MOSFET, which contains a surface layer of the same doping type as the source and drain (opposite type to the semiconductor body of the device). As a result, it has a built-in or normally on channel from source to drain with a conductance that is reduced when the gate depletes the majority carriers.

In contrast, the true MOSFET is an *enhancement-mode* or *normally off* device. The device is normally off because the body forms *p–n* junctions with both the source and the drain, so no majority-carrier current can flow between them. Instead, *minority-carrier* current can flow, provided minority carriers are available. As discussed later, for gate biases that are sufficiently attractive, above **threshold**, minority

© 2006 by Taylor & Francis Group, LLC

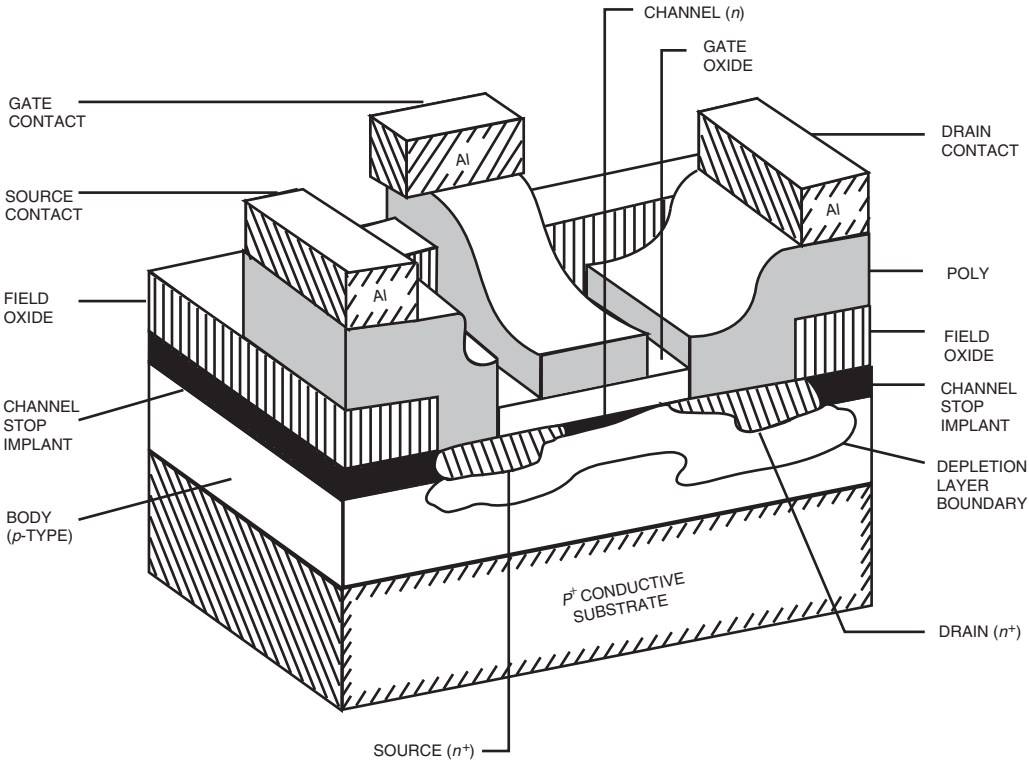


FIGURE 4.1 A high-performance *n*-channel MOSFET. The device is isolated from its neighbors by a surrounding thick *field oxide* under which is a heavily doped *channel stop* implant intended to suppress accidental channel formation that could couple the device to its neighbors. The drain contacts are placed over the field oxide to reduce the capacitance to the body, a parasitic that slows response times. These structural details are described later. (*Source:* After Brews, J.R. 1990. The submicron MOSFET. In *High-Speed Semiconductor Devices*, ed. S.M. Sze, pp. 139–210. Wiley, New York.)

carriers are drawn into a surface channel, forming a conducting path from source to drain. The gate and channel then form two sides of a capacitor separated by the gate insulator. As additional attractive charges are placed on the gate side, the channel side of the capacitor draws a balancing charge of minority carriers from the source and the drain. The more charges on the gate, the more populated the channel, and the larger the conductance. Because the gate *creates* the channel, to insure electrical continuity the gate must extend over the entire length of the separation between source and drain.

The MOSFET channel is created by attraction to the gate and relies on the insulating layer between the channel and the gate to prevent leakage of minority carriers to the gate. As a result, MOSFETs can be made only in material systems that provide very good gate insulators, and the best system known is the silicon-silicon dioxide combination. This requirement for a good gate insulator is not as important for JFETs and MESFETs where the role of the gate is to *push away* majority carriers, rather than to *attract* minority carriers. Thus, in GaAs systems where good insulators are incompatible with other device or fabricational requirements, MESFETs are used.

A more recent development in GaAs systems is the heterostructure field-effect transistor (HFET) (Pearton and Shah, 1990) made up of layers of varying compositions of Al, Ga, and As or In, Ga, P, and As. These devices are made using molecular beam epitaxy or by organometallic vapor phase epitaxy. HFETs include a variety of structures, the best known of which is the modulation doped FET (MODFET). HFETs are field-effect devices, not MOSFETs, because the gate simply modulates the carrier density in a pre-existent channel between ohmic contacts. The channel is formed spontaneously, regardless of the quality of the gate insulator, as a condition of equilibrium between the layers, just as a depletion layer is

© 2006 by Taylor & Francis Group, LLC

formed in a *p–n* junction. The resulting channel is created very near to the gate electrode, resulting in gate control as effective as in a MOSFET.

The silicon-based MOSFET has been successful primarily because the silicon-silicon dioxide system provides a stable interface with low trap densities and because the oxide is impermeable to many environmental contaminants, has a high breakdown strength, and is easy to grow uniformly and reproducibly (Nicollian and Brews, 1982). These attributes allow easy fabrication using lithographic processes, resulting in integrated circuits (ICs) with very small devices, very large device counts, and very high reliability at low cost. Because the importance of the MOSFET lies in this relationship to high-density manufacture, an emphasis of this chapter is to describe the issues involved in continuing miniaturization.

An additional advantage of the MOSFET is that it can be made using either electrons or holes as channel carrier. Using both types of devices in so-called complementary MOS (CMOS) technology allows circuits that draw no DC power if current paths include at least one series connection of both types of device because, in steady state, only one or the other type conducts, not both at once. Of course, in exercising the circuit, power is drawn during switching of the devices. This flexibility in choosing *n*- or *p*-channel devices has enabled large circuits to be made that use low-power levels. Hence, complex systems can be manufactured without expensive packaging or cooling requirements.

4.2 Current-Voltage Characteristics

The derivation of the current-voltage characteristics of the MOSFET can be found in (Annaratone, 1986; Brews, 1981; and Pierret, 1990). Here a qualitative discussion is provided.

Strong-Inversion Characteristics

In Fig. 4.2 the source-drain current I_D is plotted vs. drain-to-source voltage V_D (the I–V curves for the MOSFET). At low V_D the current increases approximately linearly with increased V_D, behaving like a simple resistor with a resistance that is controlled by the gate voltage V_G: as the gate voltage is made more attractive for channel carriers, the channel becomes stronger, more carriers are contained in the channel, and its resistance R_{ch} drops. Hence, at larger V_G the current is larger.

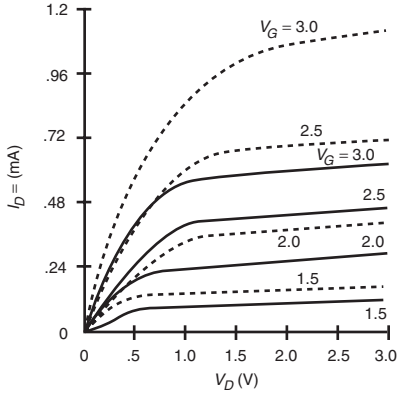


FIGURE 4.2 Drain current I_D vs. drain voltage V_D for various choices of gate bias V_G. The dashed-line curves are for a *long-channel* device for which the current in saturation increases quadratically with gate bias. The solid-time curves are for a *short-channel* device that is approaching *velocity saturation* and thus exhibits a more linear increase in saturation current with gate bias, as discussed in the text.

At large V_D the curves flatten out, and the current is less sensitive to drain bias. The MOSFET is said to be in *saturation*. There are different reasons for this behavior, depending on the field along the channel caused by the drain voltage. If the source-drain separation is short, near, or below a micrometer, the usual drain voltage is sufficient to create fields along the channel of more then a few $\times 10^4$ V/cm. In this case the carrier energy is sufficient for carriers to lose energy by causing vibrations of the silicon atoms composing the crystal (optical phonon emission). Consequently, the carrier velocity does not increase much with increased field, saturating at a value $\upsilon_{sat} \approx 10^7$ cm/s in silicon MOSFETs. Because the carriers do not move faster with increased V_D, the current also saturates.

For longer devices the current-voltage curves saturate for a different reason. Consider the potential along the insulator-channel interface, the surface potential. Whatever the surface potential is at the source end of the channel, it varies from the source end to a value larger at the drain end by V_D because the drain potential is V_D higher than the source. The gate, on the other hand, is at the same potential everywhere. Thus, the difference in potential between the gate and the source is larger than that between the gate and

© 2006 by Taylor & Francis Group, LLC

drain. Correspondingly, the oxide field at the source is larger than that at the drain and, as a result, less charge can be supported at the drain. This reduction in attractive power of the gate reduces the number of carriers in the channel at the drain end, increasing channel resistance. In short, we have $I_D \approx V_D/R_{ch}$ but the channel resistance $R_{ch} = R_{ch}(V_D)$ is increasing with V_D. As a result, the current-voltage curves do not continue along the initial straight line, but bend over and saturate.

Another difference between the current-voltage curves for short devices and those for long devices is the dependence on gate voltage. For long devices, the current level in saturation $I_{D,sat}$ increases quadratically with gate bias. The reason is that the number of carriers in the channel is proportional to $V_G - V_{TH}$ (where V_{TH} is the *threshold voltage*) as is discussed later, the channel resistance $R_{ch} \propto 1/(V_G - V_{TH})$, and the drain bias in saturation is approximately V_G. Thus, $I_{D,sat} = V_D/R_{ch} \propto (V_G - V_{TH})^2$, and we have quadratic dependence. When the carrier velocity is saturated, however, the dependence of the current on drain bias is suppressed because the speed of the carriers is fixed at υ_{sat}, and $I_{D,sat} \propto \upsilon_{sat}/R_{ch} \propto (V_G - V_{TH})\upsilon_{sat}$, a linear gate-voltage dependence. As a result, the current available from a short device is not as large as would be expected if we assumed it behaved like a long device.

Subthreshold Characteristics

Quite different current-voltage behavior is seen in **subthreshold**, that is, for gate biases so low that the channel is in *weak inversion*. In this case the number of carriers in the channel is so small that their charge does not affect the potential, and channel carriers simply must adapt to the potential set up by the electrodes and the dopant ions. Likewise, in subthreshold any flow of current is so small that it causes no potential drop along the interface, which becomes an equipotential.

As there is no lateral field to move the channel carriers, they move by diffusion only, driven by a gradient in carrier density setup because the drain is effective in reducing the carrier density at the drain end of the channel. In subthreshold the current is then independent of drain bias once this bias exceeds a few tens of millivolts, enough to reduce the carrier density at the drain end of the channel to near zero.

In short devices, however, the source and drain are close enough together to begin to share control of the potential with the gate. If this effect is too strong, a drain-voltage dependence of the subthreshold characteristic then occurs, which is undesirable because it increases the MOSFET off current and can cause a drain-bias dependent threshold voltage.

Although for a well-designed device there is no drain-voltage dependence in subthreshold, gate-bias dependence is exponential. The surface is lowered in energy relative to the semiconductor body by the action of the gate. If this *surface potential* is ϕ_S below that of the body, the carrier density is enhanced by a Boltzmann factor $\exp(q\ \phi_S/kT)$ relative to the body concentration, where kT/q is the thermal voltage, $\approx$25 mV at 290 K. As ϕ_S is roughly proportional to V_G, this exponential dependence on ϕ_S leads to an exponential dependence on V_G for the carrier density and, hence, for the current in subthreshold.

4.3 Important Device Parameters

A number of MOSFET parameters are important to the performance of a MOSFET. In this section some of these parameters are discussed, particularly from the viewpoint of digital ICs.

Threshold Voltage

The threshold voltage is vaguely defined as the gate voltage V_{TH} at which the channel begins to form. At this voltage devices begin to switch from off to on, and circuits depend on a voltage swing that straddles this value. Thus, threshold voltage helps in deciding the necessary supply voltage for circuit operation and it also helps in determining the leakage or off current that flows when the device is in the off state.

We now will make the definition of threshold voltage precise and relate its magnitude to the doping profile inside the device, as well as other device parameters such as oxide thickness and flatband voltage.

Threshold voltage is controlled by oxide thickness d and by body doping. To control the body doping, ion implantation is used, so that the dopant-ion density is not simply a uniform extension of the bulk, background level N_B ions/unit volume, but has superposed on it an implanted ion density. To estimate

© 2006 by Taylor & Francis Group, LLC

the threshold voltage, we need a picture of what happens in the semiconductor under the gate as the gate voltage is changed from its off level toward threshold.

If we imagine changing the gate bias from its off condition toward threshold, at first the result is to repel majority carriers, forming a surface *depletion layer,* refer to Fig. 4.1. In the depletion layer there are almost no carriers present, but there are dopant ions. In n-type material these dopant ions are positive donor impurities that cannot move under fields because they are locked in the silicon lattice, where they have been deliberately introduced to replace silicon atoms. In p-type material these dopant ions are negative acceptors. Thus, each charge added to the gate electrode to bring the gate voltage closer to threshold causes an increase in the depletion-layer width sufficient to balance the gate charge by an equal but opposite charge of dopant ions in the silicon depletion layer.

This expansion of the depletion layer continues to balance the addition of gate charge until threshold is reached. Then this charge response changes: above threshold any additional gate charge is balanced by an increasingly strong inversion layer or channel. The border between a depletion-layer and an inversion-layer response, threshold, should occur when

$$\frac{dq\,N_{\text{inv}}}{d\phi_S} = \frac{dQ_D}{d\phi_S} \tag{4.1}$$

where $d\phi_S$ isthe small changein surfacepotential that corresponds to our incremental changein gate charge, $q\,N_{\text{inv}}$ is the inversion-layer charge/unit area, and Q_D the depletion-layer charge/unit area. According to Eq. (4.1), the two types of response are equal at threshold, so that one is larger than the other on either side of this condition. To be more quantitative, the rate of increase in $q\,N_{\text{inv}}$ is exponential, that is, its rate of change is proportional to $q\,N_{\text{inv}}$, and so as $q\,N_{\text{inv}}$ increases, so does the left side of Eq. (4.1). On the other hand, Q_D has a square-root dependence on ϕ_S, which means its rate of change becomes smaller as Q_D increases. Thus, as surface potential is increased, the left side of Eq. (4.1) increases $\propto q\,N_{\text{inv}}$ until, at threshold, Eq. (4.1) is satisfied. Then, beyond threshold, the exponential increase in $q\,N_{\text{inv}}$ with ϕ_S swamps Q_D, making change in $q\,N_{\text{inv}}$ the dominant response. Likewise, below threshold, the exponential decrease in $q\,N_{\text{inv}}$ with decreasing ϕ_S makes $q\,N_{\text{inv}}$ negligible and change in Q_D becomes the dominant response. The abruptness of this change in behavior is the reason for the term threshold to describe MOSFET switching.

To use Eq. (4.1) to find a formula for threshold voltage, we need expressions for N_{inv} and Q_D. Assuming the interface is held at a lower energy than the bulk due to the charge on the gate, the minority-carrier density at the interface is larger than in the bulk semiconductor, even below threshold. Below threshold and even up to the threshold of Eq. (4.1), the number of charges in the channel/unit area N_{inv} is given for n-channel devices approximately by (Brews, 1981)

$$N_{\text{inv}} \approx d_{\text{INV}} \frac{n_i^2}{N_B} e^{q(\phi_S - V_S)/kT} \tag{4.2}$$

where the various symbols are defined as follows: n_i is the intrinsic carrier density/unit volume $\approx 10^{10}/\text{cm}^3$ in silicon at 290 K and V_S is the body reverse bias, if any. The first factor, d_{INV}, is an effective depth of minority carriers from the interface given by

$$d_{\text{INV}} = \frac{\varepsilon_s kT/q}{Q_D} \tag{4.3}$$

where Q_D is the depletion-layer charge/unit area due to charged dopant ions in the region where there are no carriers and ε_s is the dielectric permittivity of the semiconductor.

Equation (4.2) expresses the net minority-carrier density/unit area as the product of the bulk minority-carrier density/unit volume n_i^2/N_B, with the depth of the minority-carrier distribution d_{INV} multiplied in turn by the customary Boltzmann factor $\exp(q(\phi_S - V_S)/kT)$ expressing the enhancement of the interface density over the bulk due to lower energy at the interface. The depth d_{INV} is related to the carrier distribution near the interface using the approximation (valid in weak inversion) that the minority-carrier density decays exponentially with distance from the oxide-silicon surface. In this approximation, d_{INV}

© 2006 by Taylor & Francis Group, LLC

is the *centroid* of the minority-carrier density. For example, for a uniform bulk doping of 10^{16} dopant ions/cm^3 at 290 K, using Eq. (4.2) and the surface potential at threshold from Eq. (4.7) ($\phi_{\rm TH} = 0.69$ V), there are $Q_D/q = 3 \times 10^{11}$ charges/cm^2 in the depletion layer at threshold. This Q_D corresponds to a $d_{\rm INV} = 5.4$ nm and a carrier density at threshold of $N_{\rm inv} = 5.4 \times 10^9$ charges/cm^2.

The next step in using the definition of threshold, Eq. (4.1), is to introduce the depletion-layer charge/unit area Q_D. For the ion-implanted case, Q_D is made up of two terms (Brews, 1981)

$$Q_D = q N_B L_B (2(q\phi_{\rm TH}/kT - m_1 - 1))^{\frac{1}{2}} + q D_I \tag{4.4}$$

where the first term is Q_B, the depletion-layer charge from bulk dopant atoms in the depletion layer with a width that has been reduced by the first moment of the implant, namely, m_1 given in terms of the centroid of the implant x_C by

$$m_1 = \frac{D_I x_C}{N_B L_B^2} \tag{4.5}$$

The second term is the additional charge due to the implanted-ion density within the depletion layer, D_I/unit area. The Debye length L_B is defined as

$$L_B^2 \equiv \left[\frac{kT}{q}\right]\left[\frac{\varepsilon_s}{q N_B}\right] \tag{4.6}$$

where ε_s is the dielectric permittivity of the semiconductor. The Debye length is a measure of how deeply a variation of surface potential penetrates into the body when $D_I = 0$ and the depletion layer is of zero width.

Approximating $q N_{\rm inv}$ by Eq. (4.2) and Q_D by Eq. (4.7), Eq. (4.1) determines the surface potential at threshold $\phi_{\rm TH}$ to be

$$\phi_{\rm TH} = 2(kT/q)\ell n(N_B/n_i) + (kT/q)\ell n\left[1 + \frac{q D_I}{Q_B}\right] \tag{4.7}$$

where the new symbols are defined as follows: Q_B is the depletion-layer charge/unit area due to bulk body dopant N_B in the depletion layer, and $q D_I$ is the depletion-layer charge/unit area due to implanted ions in the depletion layer between the inversion-layer edge and the depletion-layer edge. Because even a small increase in ϕ_S above $\phi_{\rm TH}$ causes a large increase in $q N_{\rm inv}$, which can balance a rather large change in gate charge or gate voltage, ϕ_S does not increase much as $V_G - V_{\rm TH}$ increases. Nonetheless, in strong inversion $N_{\rm inv} \approx 10^{12}$ charges/cm^2, and so in **strong inversion** ϕ_S will be about $7kT/q$ larger than $\phi_{\rm TH}$.

Equation (4.7) indicates for uniform doping (no implant, $D_I = 0$) that threshold occurs approximately for $\phi_S = \phi_{\rm TH} = 2(kT/q)\ell n(N_B/n_i) \equiv 2\phi_B$, but for the nonuniformly doped case, a larger surface potential is needed, assuming the case of a normal implant where D_I is positive, increasing the dopant density. The implant increases the required surface potential because the field at the surface is larger, narrowing the inversion layer and reducing the channel strength for $\phi_S = 2\phi_B$. Hence, a somewhat larger surface potential is needed to increase $q N_{\rm inv}$ to the point that Eq. (4.1) is satisfied. Equation (4.1) would not apply if a significant fraction of the implant were confined to lie within the inversion layer itself. No realistic implant can be confined within a distance comparable to an inversion-layer thickness (a few tens of nanometers), however, and so Eq. (4.7) covers practical cases.

With the surface potential $\phi_{\rm TH}$ known, the potential on the gate at threshold $\Phi_{\rm TH}$ can be found if we know the oxide field $F_{\rm ox}$ by simply adding the potential drop across the semiconductor to that across the oxide. That is, $\Phi_{\rm TH} = \phi_{\rm TH} + F_{\rm ox} d$, where d is the oxide thickness and $F_{\rm ox}$ is given by Gauss' law as

$$\varepsilon_{\rm ox} F_{\rm ox} = Q_D \tag{4.8}$$

There are two more complications in finding the threshold voltage. First, the gate *voltage* $V_{\rm TH}$ usually differs from the gate *potential* $\Phi_{\rm TH}$ at threshold because of a work-function difference between the body and the gate material. This difference causes a spontaneous charge exchange between the two materials as

© 2006 by Taylor & Francis Group, LLC

soon as the MOSFET is placed in a circuit allowing charge transfer to occur. Thus, even before any voltage is applied to the device, a potential difference exists between the gate and the body due to spontaneous charge transfer. The second complication affecting threshold voltage is the existence of charges in the insulator and at the insulator–semiconductor interface. These nonideal contributions to the overall charge balance are due to traps and fixed charges incorporated during the device processing.

Ordinarily interface-trap charge is negligible ($< 10^{10}/\text{cm}^2$ in silicon MOSFETs) and the other nonideal effects on threshold voltage are accounted for by introducing the *flatband voltage* V_{FB}, which corrects the gate bias for these contributions. Then, using Eq. (4.8) with $F_{\text{ox}} = (V_{\text{TH}} - V_{\text{FB}} - \phi_{\text{TH}})/d$ we find

$$V_{\text{TH}} = V_{\text{FB}} + \phi_{\text{TH}} + Q_D \frac{d}{\varepsilon_{\text{ox}}} \tag{4.9}$$

which determines V_{TH} even for the nonuniformly doped case, using Eq. (4.7) for ϕ_{TH} and Q_D at threshold from Eq. (4.4). If interface-trap charge/unit area is not negligible, then terms in the interface-trap charge/unit area Q_{IT} must be added to Q_D in Eq. (4.9).

From Eqs. (4.4) and (4.7), the threshold voltage depends on the implanted dopant-ion profile only through two parameters, the net charge introduced by the implant in the region between the inversion layer and the depletion-layer edge qD_I, and the centroid of this portion of the implanted charge x_C. As a result, a variety of implants can result in the same threshold, ranging from the extreme of a δ-function spike implant of dose D_I/unit area located at the centroid x_C, to a box type rectangular distribution with the same dose and centroid, namely, a rectangular distribution of width $x_W = 2x_C$ and volume density D_I/x_W. (Of course, x_W must be no larger than the depletion-layer width at threshold for this equivalence to hold true, and x_C must not lie within the inversion layer.) This weak dependence on the details of the profile leaves flexibility to satisfy other requirements, such as control of off current.

As already stated, for gate biases $V_G > V_{\text{TH}}$, any gate charge above the threshold value is balanced mainly by inversion-layer charge. Thus, the additional oxide field, given by $(V_G - V_{\text{TH}})/d$, is related by Gauss' law to the inversion-layer carrier density approximately by

$$\varepsilon_{\text{ox}}(V_G - V_{\text{TH}})/d \approx q N_{\text{inv}} \tag{4.10}$$

which shows that channel strength above threshold is proportional to $V_G - V_{\text{TH}}$, an approximation often used in this chapter. Thus, the switch in balancing gate charge from the depletion layer to the inversion layer causes N_{inv} to switch from an exponential gate-voltage dependence in subthreshold to a linear dependence above threshold.

For circuit analysis Eq. (4.10) is a convenient *definition* of V_{TH} because it fits current-voltage curves. If this definition is chosen instead of the charge-balance definition Eq. (4.1), then Eqs. (4.1) and (4.7) result in an *approximation* to ϕ_{TH}.

Driving Ability and $I_{D,\text{sat}}$

The driving ability of the MOSFET is proportional to the current it can provide at a given gate bias. One might anticipate that the larger this current, the faster the circuit. Here this current is used to find some response times governing MOSFET circuits.

MOSFET current is dependent on the carrier density in the channel, or on $V_G - V_{\text{TH}}$, see Eq. (4.10). For a long-channel device, driving ability depends also on channel length. The shorter the channel length L, the greater the driving ability, because the channel resistance is directly proportional to the channel length. Although it is an oversimplification, let us suppose that the MOSFET is primarily in saturation during the driving of its load. This simplification will allow a clear discussion of the issues involved in making faster MOSFETs without complicated mathematics. Assuming the MOSFET to be saturated over most of the switching period, driving ability is proportional to current in saturation, or to

$$I_{D,\text{sat}} = \frac{\varepsilon_{\text{ox}} Z \mu}{2dL}(V_G - V_{\text{TH}})^2 \tag{4.11}$$

© 2006 by Taylor & Francis Group, LLC

where the factor of two results from the saturating behavior of the I–V curves at large drain biases, and Z is the width of the channel normal to the direction of current flow. Evidently, for long devices driving ability is quadratic in $V_G - V_{TH}$ and inversely proportional to d.

The result of Eq. (4.11) holds for long devices. For short devices, as explained for Fig. 4.26, the larger fields exerted by the drain electrode cause *velocity saturation* and, as a result, $I_{D,sat}$ is given roughly by (Einspruch and Gildenblat, 1989)

$$I_{D,sat} \approx \frac{\varepsilon_{ox} Z \upsilon_{sat}}{d} \frac{(V_G - V_{TH})^2}{V_G - V_{TH} + F_{sat} L} \tag{4.12}$$

where υ_{sat} is the carrier saturation velocity, about 10^7 cm/s for silicon at 290 K, and F_{sat} is the field at which velocity saturation sets in, about 5×10^4 V/cm for electrons and not well established as $\gtrsim 10^5$ V/cm for holes in silicon MOSFETs. For Eq. (4.12) to agree with Eq. (4.11) at long L, we need $\mu \approx 2\upsilon_{sat}/F_{sat} \approx 400$ cm^2/V · s for electrons in silicon MOSFETs, which is only roughly correct. Nonetheless, we can see that for devices in the submicron channel length regime, $I_{D,sat}$ tends to become independent of channel length L and becomes more linear with $V_G - V_{TH}$ and less quadratic, see Fig. 4.2. Equation (4.12) shows that velocity saturation is significant when $(V_G - V_{TH})/L \gtrsim F_{sat}$, for example, when $L \lesssim 0.5\,\mu$m if $V_G - V_{TH} = 2.3$ V.

To relate $I_{D,sat}$ to a gate response time τ_G, consider one MOSFET driving an identical MOSFET as load capacitance. Then the current from (Eq. 4.12) charges this capacitance to a voltage V_G in a gate response time τ_G given by (Shoji, 1988)

$$\begin{aligned} \tau_G &= C_G V_G / I_{D,sat} \\ &= \left[\frac{L}{\upsilon_{sat}}\right]\left[1 + \frac{C_{par}}{C_{ox}}\right]\frac{V_G(V_G - V_{TH} + F_{sat}L)}{(V_G - V_{TH})^2} \end{aligned} \tag{4.13}$$

where C_G is the MOSFET gate capacitance $C_G = C_{ox} + C_{par}$, with $C_{ox} = \varepsilon_{ox} ZL/d$ the MOSFET oxide capacitance, and C_{par} the parasitic component of the gate capacitance (Chen, 1990). The parasitic capacitance C_{par} is due mainly to overlap of the gate electrode over the source and drain and partly to fringing-field and channel-edge capacitances. For short-channel lengths, C_{par} is a significant part of C_G, and keeping C_{par} under control as L is reduced is an objective of gate-drain alignment technology. Typically, $V_{TH} \approx V_G/4$, so that

$$\tau_G = \left[\frac{L}{\upsilon_{sat}}\right]\left[1 + \frac{C_{par}}{C_{ox}}\right]\left[1.3 + 1.8\frac{F_{sat}L}{V_G}\right] \tag{4.14}$$

Thus, on an intrinsic level, the gate response time is a multiple of the transit time of an electron from source to drain, which is L/υ_{sat} in velocity saturation. At shorter L, a linear reduction in delay with L is predicted, whereas for longer devices the improvement can be quadratic in L, depending on how V_G is scaled as L is reduced.

The gate response time is not the only delay in device switching, because the drain-body p–n junction also must charge or discharge for the MOSFET to change state (Shoji, 1988). Hence, we must also consider a *drain* response time τ_D. Following Eq. (4.13), we suppose that the drain capacitance C_D is charged by the supply voltage through a MOSFET in saturation so that

$$\tau_D = C_D V_G / I_{D,sat} = \left[\frac{C_D}{C_G}\right]\tau_G \tag{4.15}$$

Equation (4.15) suggests that τ_D will show a similar improvement to τ_G as L is reduced, provided that C_D/C_G does not increase as L is reduced. However, $C_{ox} \propto L/d$, and the major component of C_{par}, namely, the overlap capacitance contribution, leads to $C_{par} \propto L_{ovlp}/d$ where L_{ovlp} is roughly three times the length of overlap of the gate over the source or drain (Chen, 1990). Then $C_G \propto (L + L_{ovlp})/d$ and, to keep the C_D/C_G ratio from increasing as L is reduced, either C_D or oxide thickness d must be reduced along with L.

Clever design can reduce C_D. For example, various *raised-drain* designs reduce the drain-to-body capacitance by separating much of the drain area from the body using a thick oxide layer. The contribution

© 2006 by Taylor & Francis Group, LLC

to drain capacitance stemming from the sidewall depletion-layer width next to the channel region is more difficult to handle, because the sidewall depletion layer is deliberately reduced during miniaturization to avoid *short-channel* effects, that is, drain influence on the channel in competition with gate control. As a result, this sidewall contribution to the drain capacitance tends to increase with miniaturization unless junction depth can be shrunk.

Equations (4.14) and (4.15) predict reduction of response times by reduction in channel length L. Decreasing oxide thickness leads to no improvement in τ_G, but Eq. (4.15) shows a possibility of improvement in τ_D. The *ring oscillator,* a closed loop of an odd number of inverters, is a test circuit the performance of which depends primarily on τ_G and τ_D. Gate delay/stage for ring oscillators is found to be near 12 ps/stage at 0.1-μm channel length, and 60 ps/stage at 0.5 μm.

For circuits, interconnection capacitances and fan out (multiple MOSFET loads) will increase response times beyond the device response time, even when parasitics are taken into account. Thus, we are led to consider interconnection delay τ_{INT}. Although a lumped model suggests, as with Eq. (4.15), that $\tau_{\text{INT}} \approx (C_{\text{INT}}/C_G)\tau_G$, the length of interconnections requires a *distributed* model. Interconnection delay is then

$$\tau_{\text{INT}} = R_{\text{INT}}\, C_{\text{INT}}/2 + R_{\text{INT}}\, C_G + (1 + C_{\text{INT}}/C_G)\tau_G \tag{4.16}$$

where R_{INT} is the interconnection resistance, C_{INT} is the interconnection capacitance, and we have assumed that the interconnection joins a MOSFET driver in saturation to a MOSFET load, C_G. For small R_{INT}, the τ_{INT} is dominated by the last term, which resembles Eqs. (4.13) and (4.15). Unlike the ratio C_D/C_G in Eq. (4.15), however, it is difficult to reduce or even maintain the ratio C_{INT}/C_G in Eq. (4.16) as L is reduced. Remember, $C_G \propto Z(L + L_{\text{ovlp}})/d$. Reduction of L, therefore, tends to increase C_{INT}/C_G, especially because interconnect cross sections cannot be reduced without impractical increases in R_{INT}. What is worse, along with reduction in L, chip sizes usually increase, making line lengths longer, increasing R_{INT} even at constant cross section. As a result, interconnection delay becomes a major problem as L is reduced. The obvious way to keep C_{INT}/C_G under control is to increase the device width Z so that $C_G \propto Z(L + L_{\text{ovlp}})/d$ remains constant as L is reduced. A better way is to cascade drivers of increasing Z (Chen, 1990; Shoji, 1988). Either solution requires extra area, however, reducing the packing density that is a major objective in decreasing L in the first place. An alternative is to reduce the oxide thickness d, a major technology objective.

Transconductance

Another important device parameter is the small-signal transconductance g_m (Malik, 1995; Sedra and Smith, 1991; Haznedar, 1991) that determines the amount of output current swing at the drain that results from a given input voltage variation at the gate, that is, the small-signal gain

$$g_m = \left.\frac{\partial I_D}{\partial V_G}\right|_{V_D=\text{const}} \tag{4.17}$$

Using the chain rule of differentiation, the transconductance in saturation can be related to the small-signal *transition* or *unity-gain frequency* that determines at how high a frequency ω the small-signal current gain $|\iota_{\text{out}}/\iota_{\text{in}}| = g_m/(\omega C_G)$ drops to unity. Using the chain rule

$$g_m = \frac{\partial I_{D,\text{sat}}}{\partial Q_G}\frac{\partial Q_G}{\partial V_G} = \omega_T C_G \tag{4.18}$$

where C_G is the oxide capacitance of the device, $C_G = \partial Q_G/\partial V_G|_{V_D}$ where Q_G is the charge on the gate electrode. The frequency ω_T is a measure of the small-signal, high-frequency speed of the device, neglecting parasitic resistances. Using Eq. (4.12) in Eq. (4.18) we find that the transition frequency also is related to the transit time L/υ_{sat} of Eq. (4.14), so that both the digital and small-signal circuit speeds are related to this parameter.

© 2006 by Taylor & Francis Group, LLC

Output Resistance and Drain Conductance

For small-signal circuits the output resistance r_o of the MOSFET (Malik, 1995; Sedra and Smith, 1991) is important in limiting the gain of amplifiers. This resistance is related to the small-signal drain conductance in saturation by

$$r_o = \frac{1}{g_D} = \left.\frac{\partial V_D}{\partial I_{D,\text{sat}}}\right|_{V_G=\text{const}} \tag{4.19}$$

If the MOSFET is used alone as a simple amplifier with a load line set by a resistor R_L, the gain becomes

$$\left|\frac{v_o}{v_{\text{in}}}\right| = g_m \frac{R_L r_o}{R_L + r_o} \leq g_m R_L \tag{4.20}$$

showing how gain is reduced if r_o is reduced to a value approaching R_L.

As devices are miniaturized, r_o is decreased and g_D increased, due to several factors. At moderate drain biases, the main factor is channel-length modulation, the reduction of the channel length with increasing drain voltage that results when the depletion region around the drain expands toward the source, causing L to become drain-bias dependent. At larger drain biases, a second factor is drain control of the inversion-layer charge density that can compete with gate control in short devices. This is the same mechanism discussed later in the context of subthreshold behavior. At rather high drain bias, carrier multiplication further lowers r_o.

In a digital inverter, a lower r_o widens the voltage swing needed to cause a transition in output voltage. This widening increases power loss due to current spiking during the transition and reduces noise margins (Annaratone, 1986). It is not, however, a first-order concern in device miniaturization for digital applications. Because small-signal circuits are more sensitive to r_o than digital circuits, MOSFETs designed for small-signal applications cannot be made as small as those for digital applications.

4.4 Limitations on Miniaturization

A major factor in the success of the MOSFET has been its compatibility with processing useful down to very small dimensions. Today channel lengths (source-to-drain spacings) of 0.5 μm are manufacturable, and further reduction to 0.1 μm has been achieved for limited numbers of devices in test circuits, such as ring oscillators. In this section some of the limits that must be considered in miniaturization are outlined (Brews, 1990).

Subthreshold Control

When a MOSFET is in the off condition, that is, when the MOSFET is in subthreshold, the off current drawn with the drain at supply voltage must not be too large in order to avoid power consumption and discharge of ostensibly isolated nodes (Shoji, 1988). In small devices, however, the source and drain are closely spaced, and so there exists a danger of direct interaction of the drain with the source, rather than an interaction mediated by the gate and channel. In an extreme case, the drain may draw current directly from the source, even though the gate is off (*punchthrough*). A less extreme but also undesirable case occurs when the drain and gate jointly control the carrier density in the channel (*drain-induced barrier lowering*, or drain control of threshold voltage). In such a case, the on-off behavior of the MOSFET is not controlled by the gate alone, and switching can occur over a range of gate voltages dependent on the drain voltage. Reliable circuit design under these circumstances is very complicated, and testing for design errors is prohibitive. Hence, in designing MOSFETs, a drain-bias independent subthreshold behavior is necessary.

A measure of the range of influence of the source and drain is the depletion-layer width of the associated p–n junctions. The depletion layer of such a junction is the region in which all carriers have been depleted, or pushed away, due to the potential drop across the junction. This potential drop includes the applied bias across the junction and a spontaneous built-in potential drop induced by spontaneous charge exchange

© 2006 by Taylor & Francis Group, LLC

when p- and n-regions are brought into contact. The depletion-layer width W of an abrupt junction is related to potential drop V and dopant-ion concentration/unit volume N by

$$W = \left[\frac{2\varepsilon_s V}{qN}\right]^{\frac{1}{2}} \tag{4.21}$$

To avoid subthreshold problems, a commonly used rule of thumb is to make sure that the channel length is longer than a minimum length $L_{\min}$ related to the junction depth r_j, the oxide thickness d, and the depletion-layer widths of the source and drain W_S and W_D by (Brews, 1990)

$$L_{\min} = A[r_j d(W_S W_D)^2]^{\frac{1}{3}} \tag{4.22}$$

where the empirical constant $A = 0.88\ \text{nm}^{-1/3}$ if r_j, W_S, and W_D are in micrometers and d is in nanometers.

Equation (4.22) shows that smaller devices require shallower junctions (smaller r_j), or thinner oxides (smaller d), or smaller depletion-layer widths (smaller voltage levels or heavier doping). These requirements introduce side effects that are difficult to control. For example, if the oxide is made thinner while voltages are not reduced proportionately, then oxide fields increase, requiring better oxides. If junction depths are reduced, better control of processing is required, and the junction resistance is increased due to smaller cross sections. To control this resistance, various *self-aligned contact* schemes have been developed to bring the source and drain contacts closer to the gate (Brews, 1990; Einspruch and Gildenblat, 1989), reducing the resistance of these connections. If depletion-layer widths are reduced by increasing the dopant-ion density, the *driving ability* of the MOSFET suffers because the threshold voltage increases. That is, Q_D increases in Eq. (4.9), reducing $V_G - V_{\text{TH}}$. Thus, increasing V_{TH} results in slower circuits.

As secondary consequences of increasing dopant-ion density, channel conductance is further reduced due to the combined effects of increased scattering of electrons from the dopant atoms and increased oxide fields that pin carriers in the inversion layer closer to the insulator-semiconductor interface, increasing scattering at the interface. These effects also reduce driving ability, although for shorter devices they are important only in the linear region (that is, below saturation), assuming that mobility μ is more strongly affected than saturation velocity υ_{sat}.

Hot-Electron Effects

Another limit on how small a MOSFET can be made is a direct result of the larger fields in small devices. Let us digress to consider why proportionately larger voltages, and thus larger fields, are used in smaller devices. First, according to Eq. (4.14), τ_G is shortened if voltages are increased, at least so long as $V_G/L \lesssim F_{\text{sat}} \approx 5 \times 10^4$ V/cm. If τ_G is shortened this way, then so are τ_D and τ_{INT}, Eqs. (4.15) and (4.16). Thus, faster response is gained by increasing voltages into the velocity saturation region. Second, the fabricational control of smaller devices has not improved proportionately as L has shrunk, and so there is a larger percentage variation in device parameters with smaller devices. Thus, disproportionately larger voltages are needed to insure all devices operate in the circuit, to overcome this increased fabricational *noise*. Thus, to increase speed and to cope with fabricational variations, fields get larger in smaller devices.

As a result of these larger fields along the channel direction, a small fraction of the channel carriers have enough energy to enter the insulating layer near the drain. In silicon-based p-channel MOSFETs, energetic holes can be trapped in the oxide, leading to a positive oxide charge near the drain that reduces the strength of the channel, degrading device behavior. In n-channel MOSFETs, energetic electrons entering the oxide create interface traps and oxide wear out, eventually leading to gate-to-drain shorts (Pimbley et al., 1989).

To cope with these problems *drain engineering* has been tried, the most common solution being the *lightly doped drain* (Chen, 1990; Einspruch and Gildenblat, 1989; Pimbley et al., 1989; Wolf, 1995). In this design, a lightly doped extension of the drain is inserted between the channel and the drain proper. To keep the field moderate and reduce any peaks in the field, the lightly doped drain extension is designed to spread the drain-to-channel voltage drop as evenly as possible. The aim is to smooth out the field at a value close to F_{sat} so that energetic carriers are kept to a minimum. The expense of this solution is an increase in

© 2006 by Taylor & Francis Group, LLC

drain resistance and a decreased gain. To increase packing density, this lightly doped drain extension can be stacked vertically alongside the gate, rather than laterally under the gate, to control the overall device area.

Thin Oxides

According to Eq. (4.22), thinner oxides allow shorter devices and, therefore, higher packing densities for devices. In addition, driving ability is increased, shortening response times for capacitive loads, and output resistance and transconductance are increased. There are some basic limitations on how thin the oxide can be made. For instance, there is a maximum oxide field that the insulator can withstand. It is thought that the intrinsic break-down voltage of SiO_2 is of the order of 10^7 V/cm, a field that can support $\approx 2 \times 10^{13}$ charges/cm^2, a large enough value to make this field limitation secondary. Unfortunately, as they are presently manufactured, the intrinsic breakdown of MOSFET oxides is much less likely to limit fields than defect-related leakage or breakdown, and control of these defects has limited reduction of oxide thicknesses in manufacture to about 5 nm to date.

If defect-related problems could be avoided, the thinnest useful oxide would probably be about 3 nm, limited by direct tunneling of channel carriers to the gate. This tunneling limit is not well established and also is subject to oxide-defect enhancement due to tunneling through intermediate defect levels. Thus, defect-free manufacture of thin oxides is a very active area of exploration.

Dopant-Ion Control

As devices are made smaller, the precise positioning of dopant inside the device is critical. At high temperatures during processing, dopant ions can move. For example, source and drain dopants can enter the channel region, causing position-dependence of threshold voltage. Similar problems occur in isolation structures that separate one device from another (Primbley et al., 1989; Einspruch and Gildenblat, 1989; Wolf, 1995).

To control these thermal effects, process sequences are carefully designed to limit high-temperature steps. This design effort is shortened and improved by the use of computer modeling of the processes. Dopant-ion movement is complex, however, and its theory is made more difficult by the growing trend to use *rapid thermal processing* that involves short-time heat treatments. As a result, dopant response is not steady state, but transient. Computer models of transient response are primitive, forcing further advance in small device design to be more empirical.

Other Limitations

Besides limitations directly related to the MOSFET, there are some broader difficulties in using MOSFETs of smaller dimension in chips involving even greater numbers of devices. Already mentioned is the increased delay due to interconnections that are lengthening due to increasing chip area and increasing complexity of connection. The capacitive loading of MOSFETs that must drive signals down these lines can slow circuit response, requiring extra circuitry to compensate.

Another limitation is the need to isolate devices from each other (Brews, 1990; Chen, 1990, Einspruch and Gildenblat, 1989; Pimbley et al., 1989; Wolf, 1995), so that their actions remain uncoupled by parasitics. As isolation structures are reduced in size to increase device densities, new parasitics are discovered. A solution to this problem is the manufacture of circuits on insulating substrates, *silicon-on-insulator* technology (Colinge, 1991). To succeed, this approach must deal with new problems, such as the electrical quality of the underlying silicon-insulator interface, and the defect densities in the silicon layer on top of this insulator.

Acknowledgments

The author is pleased to thank R.D. Schrimpf and especially S.L. Gilbert for suggestions that clarified the manuscript.

© 2006 by Taylor & Francis Group, LLC

Defining Terms

Channel: The conducting region in a MOSFET between source and drain. In an *enhancement*-mode, or normally off MOSFET the channel is an inversion layer formed by attraction of minority carriers toward the gate. These carriers form a thin conducting layer that is prevented from reaching the gate by a thin *gate-oxide* insulating layer when the gate bias exceeds *threshold*. In a *buried-channel* or *depletion*-mode, or normally on MOSFET, the channel is present even at zero gate bias, and the gate serves to increase the channel resistance when its bias is nonzero. Thus, this device is based on majority-carrier modulation, like a MESFET.

Gate: The control electrode of a MOSFET. The voltage on the gate capacitively modulates the resistance of the connecting *channel* between the *source* and *drain.*

Source, drain: The two output contacts of a MOSFET, usually formed as *p–n* junctions with the *substrate* or *body* of the device.

Strong inversion: The range of gate biases corresponding to the on condition of the MOSFET. At a fixed gate bias in this region, for low drain-to-source biases the MOSFET behaves as a simple gate-controlled resistor. At larger drain biases, the channel resistance can increase with drain bias, even to the point that the current *saturates,* or becomes independent of drain bias.

Substrate or body: The portion of the MOSFET that lies between the *source* and *drain* and under the *gate*. The gate is separated from the body by a thin *gate insulator*, usually silicon dioxide. The gate modulates the conductivity of the body, providing a gate-controlled resistance between the source and drain. The body is sometimes DC biased to adjust overall circuit operation. In some circuits the body voltage can swing up and down as a result of input signals, leading to body-effect or back-gate bias effects that must be controlled for reliable circuit response.

Subthreshold: The range of gate biases corresponding to the off condition of the MOSFET. In this regime the MOSFET is not perfectly off, but conducts a leakage current that must be controlled to avoid circuit errors and power consumption.

Threshold: The gate bias of a MOSFET that marks the boundary between on and off conditions.

References

The following references are not to the original sources of the ideas discussed in this article, but have been chosen to be generally useful to the reader.

Annaratone, M. 1986. *Digital CMOS Circuit Design*. Kluwer Academic, Boston, MA.

Brews, J.R. 1981. Physics of the MOS Transistor. In *Applied Solid State Science, Supplement 2A,* ed. D. Kahng, pp. 1–20. Academic Press, New York.

Brews, J.R. 1990. The Submicron MOSFET. In *High-Speed Semiconductor Devices*, ed. S.M. Sze, pp. 139–210. Wiley, New York.

Chen, J.Y. 1990. *CMOS Devices and Technology for VLSI*. Prentice-Hall, Englewood Cliffs, NJ.

Colinge, J.-P. 1991. *Silicon-on-Insulator Technology: Materials to VLSI*. Kluwer Academic, Boston, MA.

Haznedar, H. 1991. *Digital Microelectronics*. Benjamin/Cummings, Redwood City, CA.

Hollis, M.A. and Murphy, R.A. 1990. Homogeneous Field-Effect Transistors. In *High-Speed Semiconductor Devices*, ed. S.M. Sze, pp. 211–282. Wiley, New York.

Einspruch, N.G. and Gildenblat, G.S., eds. 1989. *Advanced MOS Device Physics,* Vol. 18, *VLSI Microstructure Science*. Academic, New York.

Malik, N.R. 1995. *Electronic Circuits: Analysis, Simulation, and Design*. Prentice-Hall, Englewood Cliffs, NJ.

Nicollian, E.H. and Brews, J.R. 1982. *MOS Physics and Technology,* Chap. 1. Wiley, New York.

Pearton, S.J. and Shah, N.J. 1990. Heterostructure Field-Effect Transistors. In *High-Speed Semiconductor Devices*, ed. S.M. Sze, pp. 283–334. Wiley, New York.

Pierret, R.F. 1990. *Field Effect Devices*, 2nd ed., Vol. 4, *Modular Series on Solid State Devices*. Addison-Wesley, Reading, MA.

Pimbley, J.M., Ghezzo, M., Parks, H.G., and Brown, D.M. 1989. *Advanced CMOS Process Technology*, ed. N.G. Einspruch, Vol. 19, *VLSI Electronics Microstructure Science*. Academic Press, New York.

© 2006 by Taylor & Francis Group, LLC

Sedra, S.S. and Smith, K.C. 1991. *Microelectronic Circuits*, 3rd ed. Saunders College Publishing, Philadelphia, PA.

Shoji, M. 1988. *CMOS Digital Circuit Technology*. Prentice-Hall, Englewood Cliffs, NJ.

Wolf, S. 1995. *Silicon Processing for the VLSI Era: Volume 3—The Submicron MOSFET*. Lattice Press, Sunset Beach, CA.

Further Information

The references given in this section have been chosen to provide more detail than is possible to provide in the limited space of this article. In particular, Annaratone (1986) and Shoji (1988) provide much more detail about device and circuit behavior. Chen (1990), Pimbley (1989), and Wolf (1995) provide many technological details of processing and its device impact. Haznedar (1991), Sedra and Smith (1991), and Malik (1995) provide much information about circuits. Brews (1981) and Pierret (1990) provide good discussion of the derivation of the device current-voltage curves and device behavior in all bias regions.

© 2006 by Taylor & Francis Group, LLC

5 Integrated Circuits

5.1 Introduction 5-1
5.2 High-Speed Design Techniques 5-1
Optimization of Gate Level Design • Clocks and Clock Schemes in High-Speed Circuit Design • Asynchronous Circuits and Systems • Interconnect Parasitics and Their Impact on High-Speed Design

Tom Chen

5.1 Introduction

Transistors and their fabrication into *very large scale integrated (VLSI)* circuits are the invention that has made modern computing possible. Since its inception, integrated circuits have been advancing rapidly from a few transistors on a small silicon die in the early 1960s to 4 millions of transistors integrated on to a single large silicon substrate. The dominant type of transistor used in today's integrated circuits is the metal-oxide-semiconductor (MOS) type transistor. The rapid technological advances in integrated circuit (IC) technology accelerated during and after the 1980s, and one of the most influential factors for such a rapid advance is the technology scaling, that is, the reduction in MOS transistor feature sizes. The MOS feature size is typically measured by the MOS transistor channel length. The smaller the transistors, the more dense the integrated circuits in terms of the number of transistors packed on to a unit area of silicon substrate, and the faster the transistor can switch. Not only can we pack more transistors onto a unit silicon area, the chip size has also increase. As the transistor gets smaller and silicon chip size gets bigger, the transistor's driving capability decreases and the interconnect parasitics (interconnect capacitance and resistance) increases. Consequently, the entire VLSI system has to be designed very carefully to meet the speed demands of the future. Common design issues include optimal gate design and transistor sizing, minimization of clock skew and proper timing budgeting, and realistic modeling of interconnect parasitics.

5.2 High-Speed Design Techniques

A modern VLSI device typically consists of several *megacells*, such as memory blocks and data-path arithmetic blocks, and a lot of basic MOS logic gates, such as inverters and NAND/NOR gates. **Complementary MOS (CMOS)** is one of the most widely used logic families, mainly because of its low-power consumption and high-noise margin. Other logic families include NMOS and PMOS logic. Because of its popularity, only the CMOS logic will be discussed. Many approaches to high-speed design discussed here are equally applicable to other logic families.

Optimizing a VLSI device for high-speed operation can be carried out at the system level, as well as at the circuit and logic level. To achieve the maximum operating speed at the circuit and logic levels for a given technology, it is essential to properly set the size of each transistor in a logic gate to optimally drive the output load. If the output load is very large, a string of drivers with geometrically increasing sizes is needed. The size of transistors in a logic gate is also determined by the impact of the transistors as a load to be driven by their preceding gates.

© 2006 by Taylor & Francis Group, LLC

Optimization of Gate Level Design

To optimize the gate level design, let us look at the performance of a single CMOS inverter as shown in Fig. 5.1. Delay of a gate is typically defined as the time difference between input transition and output transition at 50% of supply voltage. The inverter gate delay can be analytically expressed as

$$T_d = C_l(A_n/\beta_n + A_p/\beta_p)/2$$

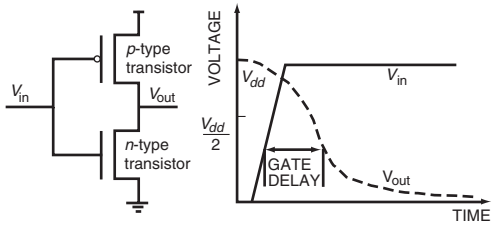


FIGURE 5.1 Gate delay in a single inverter.

where C_l is the load capacitance of the inverter; β_n and β_p are the forward current gains of n-type and p-type transistors, respectively, and are proportional to the transistor's channel width and inversely proportional to the transistor's channel length; A_n and A_p are process related parameters for a given supply voltage and they are determined by

$$A_n = [2n/(1-n) + \ell n((2(1-n) - V_0)/V_0)][V_{dd}(1-n)]$$

$$A_n = [-2p/(1+p) + \ell n((2(1+p) - V_0)/V_0)][V_{dd}(1+p)]$$

where $n = V_{thn}/V_{dd}$ and $p = V_{thp}/V_{dd}$. V_{thn} and V_{thp} are gate threshold voltages for n-channel and p-channel transistors, respectively. This expression does not take the input signal slope into account. Otherwise, the expression would become more complicated. For more complex CMOS gates, an equivalent inverter structure is constructed to reflect the effective strength of their p-tree and n-tree in order to apply the inverter delay model.

In practice, CMOS gate delay is treated in a simple fashion. The delay of a logic gate can be divided into two parts: the intrinsic delay D_{ins}, and the load-related delay D_{load}. The gate intrinsic delay is determined by the internal characteristics of the gate including the implementing technology, the gate structure, and the transistor sizes. The load-related delay is a function of the total load capacitance at the gate's output. The total gate delay can be expressed as

$$T_d = D_{\text{ins}} + C_l^* S$$

where C_l is the total load capacitance and S is the factor for gate's driving strength. $C_l^* S$ represents the gate's load-related delay. In most CMOS circuits using leading-edge submicron technologies, the total delay of a gate can be dominated by the load-related delay. For an inverter in a modern submicron CMOS technology of around 0.5-μm feature size, D_{ins} can range from 0.08 to 0.12 ns and S can range from 0.00065 to 0.00085 ns/fF depending on specifics in the technology and the minimum transistor feature size. For other more complex gates such as NAND and NOR gates, D_{ins} and S generally increase.

To optimize a VLSI circuit for its maximum operating speed, **critical paths** must be identified. A critical path in a circuit is a signal path with the longest time delay from a primary input to a primary output. The time delay on the critical path in a circuit determines the maximum operating speed of the circuit. The time delay of a critical path can be minimized by altering the size of the transistors on the critical path. Using the lumped resistor-capacitor (RC) delay model, the problem of transistor sizing can be formulated to an optimization problem with a convex relationship between the path delay and the sizes of the transistors on the path. This optimization problem is simple to solve. The solutions often have 20–30% deviation, however, compared to the **SPICE** simulation results. Realistic modeling of gate delay taking some second-order variables, such as input signal slope, into consideration has shown that the relationship between the path delay and the sizes of the transistors on the path is not convex. Such detailed analysis led to more sophisticated transistor sizing algorithms. One of these algorithms suggested using genetic methods to search for an optimal solution and has shown some promising results.

Clocks and Clock Schemes in High-Speed Circuit Design

Most of the modern electronic systems are **synchronous systems.** The clock is a central pace setter in a synchronous system to step the desired system operations through various stages of the computation.

© 2006 by Taylor & Francis Group, LLC

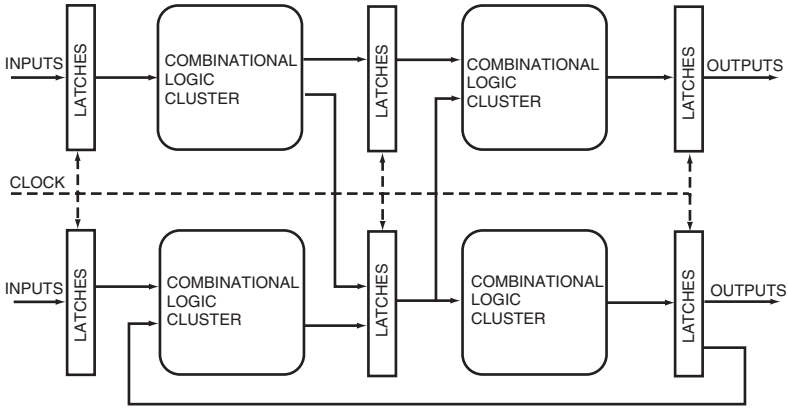


FIGURE 5.2 A typical synchronous circuit with combinational logic clusters and latches.

Latches are often used to facilitate catching the output data at the end of each clock cycle. Figure 5.2 shows the typical synchronous circuit with random logic clusters as computational blocks and latches as pace setting devices. When there exist feedbacks, as shown in Fig. 5.2, the circuit is referred to as *sequential circuit.*

A latch is also called a *register* or a *flip-flop*. The way a latch catches data depends on how it is triggered by the clock signal. Generally, there are level-triggered and edge-triggered latches, the former can be further subdivided according to the triggering polarity as positive or negative level or edge-triggered latches. The performance of a digital circuit is often determined by the maximum clock frequency the circuit can run. For a synchronous digital circuit to function properly, the longest delay through any combinational clusters must be less than the clock cycle period. Therefore, the following needs to be done for high-speed design:

- Partition the entire system so that the delays of all of the combinational clusters are as balanced as possible.
- Design the circuit of the combinational clusters so that the delay of critical paths in the circuit is minimized and less than the desired clock cycle period.
- Use a robust clock scheme to ensure that the entire system is free of race conditions and has minimal tolerable clock skew.

The first item listed is beyond the scope of this section. The second item was discussed in the preceding subsection.

Race conditions typically occur with the use of level triggered latches. Figure 5.3 shows a typical synchronous system based on level triggered latches. Because of delays on the clock distribution network, such as buffers and capacitive parasitics on the interconnect, the timing difference caused by such a distribution delay is often referred to as **clock skew** which is modeled by a delay element in Fig. 5.3. For the system to operate properly, at the positive edge each latch is supposed to capture its input data from the previous clock cycle. If the clock skew is severe shown as skewed clock clk'', however, it could be possible that the delay from $Q1$ to $D2$ becomes short enough that $D1$ is not only caught by the clk, but also caught by clk'. The solution to such a race condition caused by severe clock skew is as follows:

- Change the latch elements from level triggered to edge triggered or pseudoedge triggered latches such as latches using two-phase, nonoverlapping clocks.
- Resynthesize the system to balance to the critical path delay of different combinational clusters.
- Reduce the clock skew.

Clock skew can also cause other types of circuit malfunction if it is not attended properly. In dynamic logic, severe clock skew may result in different functional blocks being in different stages of precharging

© 2006 by Taylor & Francis Group, LLC

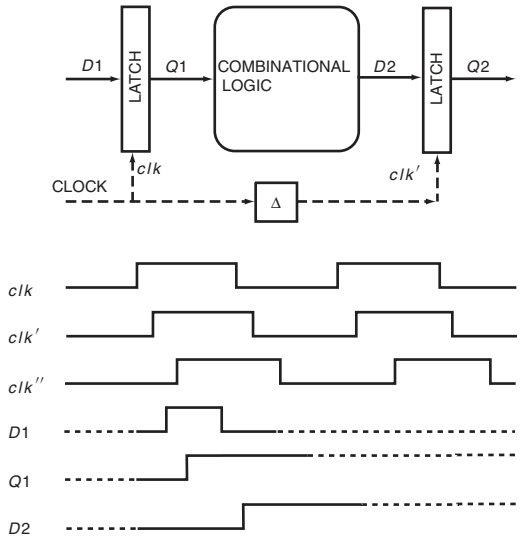


FIGURE 5.3 An example of race condition caused by a severe clock skew in the system.

or evaluating. Severe clock skew eats away a significant amount of precious cycle time in VLSI systems. Therefore, reducing clock skew is an important problem in high-speed circuit design.

As discussed earlier, clock skew is mainly caused by the imbalance of the clock distribution network. Such an imbalance can be the result of distance differences from the nearest clock driver, different functional blocks driven by different clock drivers with different driving strengths, temperature difference on the same die, device characteristic differences on the same die due to process variation, etc. Two general approaches are often taken to minimize the clock skew. The first approach deals with the way the clock signal is distributed. The geometric shape of the clock distribution network is a very important attribute. Depending on the type of system operation, several popular distribution network topologies are illustrated in Fig. 5.4. Among these topologies, **H-tree** presents least amount of clock skew and, therefore, is widely used in high-performance systems.

The second approach deals with employing additional on-chip circuits to force the clock signals of two different functional blocks to be aligned, or to force the on-chip clock signal to be aligned with the global clock signal at the system level. The widely used circuits for this purpose include **phase-locked loop** (**PLL**) and **delay-locked loop** (**DLL**). Figure 5.5 shows a simple phase-locked loop. A simple PLL consists of four components: a digital phase detector, a charge pump, a low-pass filter, and a voltage controlled oscillator (VCO). The phase detector accepts reference clock, CLK_ref, and a skewed clock, CLK_out, and compares the phase difference of the two clocks to charge or discharge the charge pump. The low-pass filter is used to convert the phase difference between the reference frequency and the skewed frequency to a voltage level. This voltage is then fed into the VCO to reduce the difference of the reference and skewed clocks until they are locked to each other.

One of the most important design parameters for PLL is the output jitter. The output jitter is demonstrated by the random deviation of the output clock's phase from the reference clock signal. Significant peak-to-peak jitter will effectively reduce the clock period. The main contributor of the output jitter is the noise on the input of the VCO. Additional jitter can be induced by the noise on power supply rails that are common to high-speed VLSI circuits. Furthermore, acquisition time of PLL, in the several microsecond range, is often longer than desirable. This is mainly attributed to the response time of the VCO. In a typical scenario where clock skew is caused by the imbalance of the distribution network, the skewed clock often has the correct frequency. What needs to be corrected is the relative phase of the clock signals. Therefore, there is no need to have a VCO. Rather, a simple delay logic can be used to modify the clock signal's phase.

© 2006 by Taylor & Francis Group, LLC

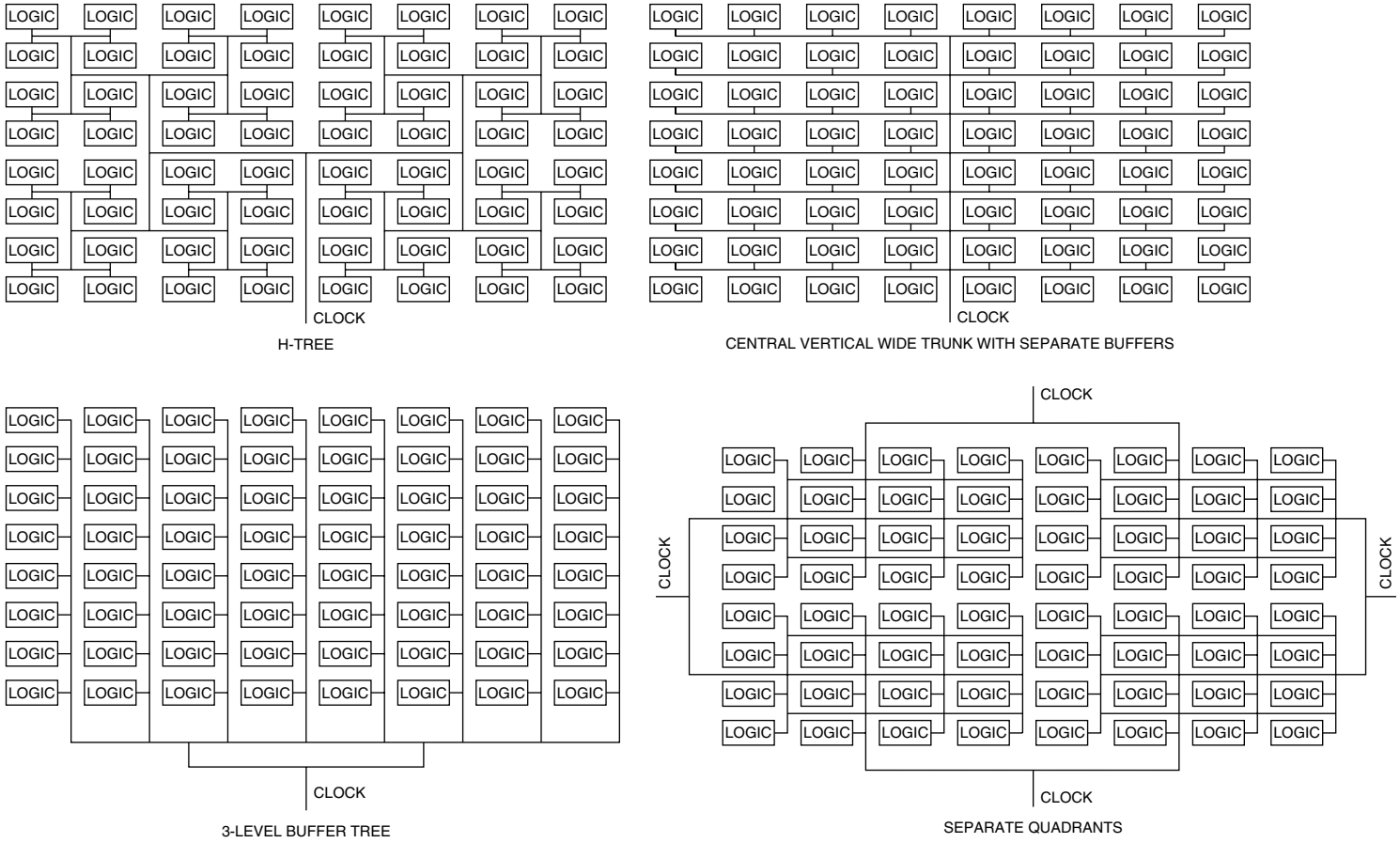


FIGURE 5.4 Various clock distribution structures.

© 2006 by Taylor & Francis Group, LLC

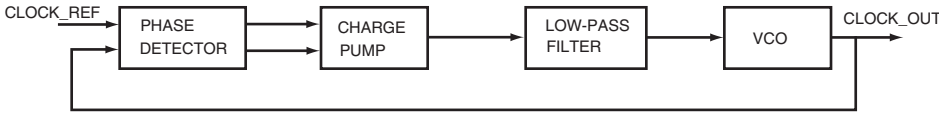


FIGURE 5.5 A simple phase-locked loop structure.

This type of simplified phase correct circuit is referred to as a delay-locked loop. By replacing the VCO with a simple programmable delay line, DLL is simpler, yet exhibits less jitter than its PLL counterpart.

Asynchronous Circuits and Systems

Clock distribution within large VLSI chips is becoming more and more of a problem for high-speed digital systems. Such a problem may be surmountable using state-of-the-art computer aided design (CAD) tools and on-chip PLL/DLL circuits. Asynchronous circuits have, nevertheless, gained a great deal of attention lately. An asynchronous circuit does not require an external clock to get it through the computation. Instead, it works on the principle of handshaking between functional blocks. Therefore, execution of a computational operation is totally dependent on the readiness of all of the input variables of the functional block. The biggest advantage of asynchronous circuits over synchronous circuits is that the correct behavior of asynchronous circuits is independent of the speed of their components or signal interconnect delays.

In a typical asynchronous circuit, functional blocks will have two more signals, *request* and *complete*, apart from their input and output signals as shown in Fig. 5.6. These two binary signals are necessary and sufficient for handshaking purposes. Even though asynchronous circuits are speed independent, the order of computation is still maintained by connecting the complete signal from one block to the request signal to another block. When the request signal is active for a functional block, indicating the computation of the preceding functional block is completed, the current functional block starts computation evaluation using its valid inputs from the preceding functional block. Once the evaluation is completed, the current functional block sets the complete signal to active to activate other functional blocks for computation. Figure 5.7 shows a schematic of such a communication protocol where block A and block B are connected in a pipeline.

To ensure that asynchronous circuits function correctly regardless of individual block speed, the *request* signal of a functional block should only be activated if the functional block has already completed the current computation. Otherwise, the current computation would be overwritten by incoming computation requests. To prevent this situation from happening, an interconnection block is required with an *acknowledge* signal from the current functional block to the preceding functional block. An active acknowledge signal indicates to the preceding function block that the current block is ready to accept new data from it. This two-way communication protocol with request and acknowledge is illustrated in Fig. 5.7. The interconnect circuit is unique to asynchronous circuits. It is often referred to as a **C-element**. Figure 5.8 shows a design of the C-element.

In recent years, much effort has been spent on applying the asynchronous circuits to real-world applications. Several totally asynchronous designs of microprocessors have demonstrated their commercial feasibility. Several issues that still need to be addressed with regard to asynchronous circuits include

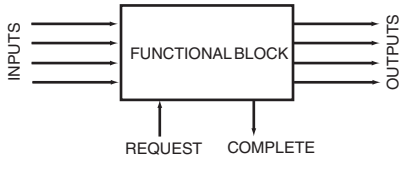


FIGURE 5.6 The single request and complete are two extra signals for a typical asynchronous circuit.

© 2006 by Taylor & Francis Group, LLC

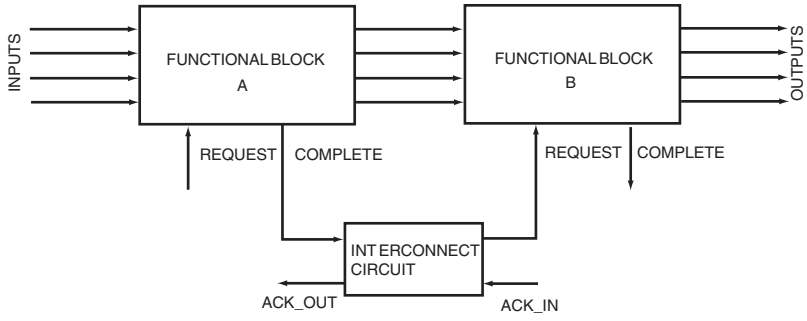


FIGURE 5.7 The schematic of a communication protocol in asynchromous systems.

acceptable amounts of silicon overhead, power efficiency, and performance as compared to their synchronous counterparts.

Interconnect Parasitics and Their Impact on High-Speed Design

On-chip interconnects present parasitic capacitance and resistance as loads to active circuits. Such parasitic loads had little impact on earlier ICs because the intrinsic gate delay dominated the total gate delay. With aggressive scaling in the VLSI process, the gate intrinsic delay decreases dramatically. The interconnect parasitics does not scale proportionally, however, and the wire resistance tends to increase, resulting in the delay caused by the interconnect load parasitics gradually becoming a dominant factor in the total gate delay. The problem is further exacerbated by the fact that when the operating speed reaches several hundred megahertz, the traditional lumped RC model is no longer accurate. It has been suggested that such a lumped RC model should be modified to include a grounded resistor and an inductor. The RLC interconnect model includes nonequilibrium initial conditions and its response waveform may be nonmonotonic. Such a model may be more accurate because the existence of the inductance reduces the rate of increase in current and, therefore, increases the signal transition time. When the operating speed increases further such that the rise time of a signal is much less than the signal transmission time from point A to point B, a transmission line model should be used. On-chip interconnect, thus, is typically modeled as a microstrip.

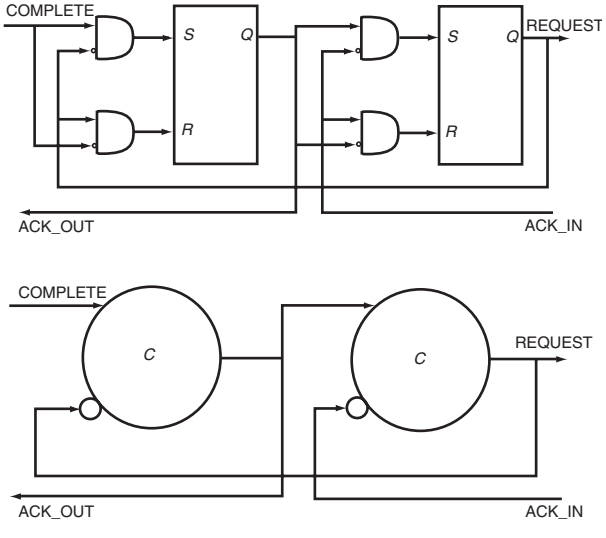


FIGURE 5.8 A schematic design of the C-element.

© 2006 by Taylor & Francis Group, LLC

TABLE 5.1 Transmission Line Velocity in Some Common Materials

	Velocity, cm/ns
Polymide	16–19
SiO_2	15
Epoxy glass (PCB)	13
Aluminium	10

The characteristics of a transmission line are determined by its relative dielectric constant and magnetic permeability. Table 5.1 shows the signal transmission velocity in some common materials used in VLSI. As a rule of thumb, the transmission line phenomena become significant when

$$t_r < 2.5^* t_f$$

where t_r is the rise time of a signal and t_f is the signal transmission time, which is the interconnect length divided by the signal traveling velocity in the given material. The interconnect can be treated as a lumped RC network when

$$t_r > 5^* t_f$$

The signal rise time depends on the driver design and the transmission line's characteristic impedance Z_0. In MOS ICs, the load device at the receiving end of the transmission line can always be treated as an open circuit. Therefore, driver design is a very important aspect of high-speed circuit design. The ideal case is to have the driver's output impedance match the transmission line's characteristic impedance. Driving an unterminated transmission line (the MOS IC case) with its output impedance lower than the line's characteristic impedance, however, can increase driver's settling time due to excess ringing and, therefore, is definitely to be avoided. Excess ringing at the receiving end could also cause the load to switch undesirably. Assuming MOS transistor's threshold is 0.6–0.8 V, to ensure that no undesirable switching takes place, the output impedance of the drive should be at least a third of the charactertstic impedance of the transmission line. When the output impedance is higher than the line's characteristic impedance, multiple wave trips of the signal may be required to switch the load. To ensure that only one wave trip is needed to switch the load, the output impedance of the driver should be within 60% of the characteristic impedance of the transmission line.

For a lossy transmission line due to parasitic resistance of on-chip interconnects, an exponential attenuating transfer function can be applied to the signal transfer at any point on the transmission line. The rate of the attenuation is proportional to the unit resistance of the interconnect. When operating frequency increases beyond a certain level, the on-chip transmission media exhibits the *skin effect* in which the time-varying currents concentrate near the skin of the conductor. Therefore, the unit resistance of the transmission media increases dramatically.

Defining Terms

Application-specific integrated circuit (ASIC): Device designed specifically for a particular application.

Application-specific standard product (ASSP): Device designed specifically for one area of applications, such as graphics and video processing.

Asynchronous system: A system in which the progress of a computation is driven by the readiness of all the necessary input variables for the computation through a handshaking protocol. Therefore, no central clock is needed.

C-element: A circuit used in an asynchronous as an interconnect circuit. The function of this circuit is to facilitate the handshaking communication protocol between two functional blocks.

Clock skew: A phase difference between two clock signals at different part of a chip/system due to imbalance of the distribution media and the distribution network.

© 2006 by Taylor & Francis Group, LLC

Complementary metal-oxide silicon (CMOS): It is a very popular integrated circuit type in use today.
Critical path: A signal path from a primary input pin to a primary output pin with the longest delay time in a logic block.
Delay-locked loop (DLL): It is similar to PLL except that it has better jitter suppression capability.
Digital signal processor (DSP): A processing device specialized in popular math routines used by signal processing algorithms.
Field programmable gate array (FPGA): A popular device which can be tailored to a particular application by loading a customizing program on to the chip.
H-tree: A popular clock distribution tree topologically that resembles the H shape. It introduces the least amount of clock skew compared to other distribution topologies.
Phase-locked loop (PLL): A circuit that can detect the phase difference of two signals and reduce the difference in the presence of the phase difference.
Programmable logic devices (PLD): A class of IC products which are easy to customize for a particular application.
SPICE: A popular circuit level simulation program to perform detailed analysis of circuit behavior.
Synchronous system: A system in which a computation is divided into unit periods defined by a central clock signal. Signal transfer within the system typically occurred at the transition edge of the clock signal.

References

Bakoglu, H.B. 1991. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, Reading, MA.

Dill, D.L. 1989. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. MIT Press, Cambridge, MA.

Gardner, F.M. 1979. *Phaselock Techniques*, 2nd ed. Wiley, New York.

Jeong, D. et al. 1987. Design of PLL-based clock generation circuits. *IEEE J. Solid-State Circuits* SC-22(2): 255–261.

Johnson, M. and Hudson, E. 1988. A variable delay line PLL for CPU-coprocessor synchronization. *IEEE J. Solid-State Circuits* (Oct.):1218–1223.

Meng, T.H. 1991. *Synchronization Design for Digital Systems*. Kluwer Academic, Norwell, MA.

Rosenstark, S. 1994. *Transmission Lines in Computer Engineering*. McGraw-Hill, New York.

Sapatnekar, S., Rao, V., and Vaidya, P. 1992. A convex optimization approach to transistor sizing for CMOS circuits. *Proc. ICCAD*, pp. 482–485.

Wang, X. and Chen, T. 1995. Performance and area optimization of VLSI systems using genetic algorithms. *Int. J. of VLSI Design* 3(1):43–51.

Weste, N. and Eshraghian, K. 1993. *Principle of CMOS VLSI Design: A Systems Perspective*, 2nd ed. Addison-Wesley, Reading, MA.

Further Information

For general information on the VLSI design process and various design issues, consult several excellent reference books, two of which are listed in the reference section, including Mead and Conway's *Introduction to VLSI Systems*, Glasser and Dobberpuhl's *The Design and Analysis of VLSI Circuits*, and Geiger's *VLSI Design Techniques for Analog and Digital Circuits*. *IEEE Journal of Solid-State Circuits* provides an excellent source for the latest development of novel and high-performance VLSI devices.

Some of the latest applications of PLLs and DLLs can be found in the *Proceedings of International Solid-State Circuit Conference*, the *Symposium on VLSI Circuits*, and the *Custom Integrated Circuit Conference*.

For information on modeling of VLSI interconnects and their transmission line treatment, consult the *Proceedings of Design Automation Conference* and *International Conference on Computer-Aided Design*. *IEEE Transactions on CAD* is also an excellent source of information on the subject.

© 2006 by Taylor & Francis Group, LLC

6

Integrated Circuit Design

6.1 Introduction 6-1
6.2 An Overview of the IC Design Process 6-2
6.3 General Considerations in IC Design 6-2
Device Scaling • Geometric Design Rules
6.4 Design of Small-Scale and Medium-Scale Integrated Circuits 6-6
NMOS Inverters • NMOS Gates • CMOS Inverters • CMOS Gates • Bipolar Gates • Medium-Scale Integrated Circuits
6.5 LSI and VLSI Circuit Design 6-12
Multiphase Clocking
6.6 Increasing Packing Density and Reducing Power Dissipation in MOS Circuits 6-14
6.7 Gate Arrays 6-15
6.8 Standard Cells 6-16
6.9 Programmable Logic Devices 6-16
Programmable Logic Array
6.10 Reducing Propagation Delays 6-19
Resistance-Capacitance (RC) Delay Lines • Superbuffers
6.11 Output Buffers 6-22

Samuel O. Agbo

Eugene D. Fabricius

6.1 Introduction

Integrated circuits (ICs) are classified according to their levels of complexity: small-scale integration (SSI), medium-scale integration (MSI), large-scale integration (LSI) and very large-scale integration (VLSI). They are also classified according to the technology employed for their fabrication (bipolar, N metal oxide semiconductor (NMOS), complementary metal oxide semiconductor (CMOS), etc.). The design of integrated circuits needs to be addressed at the SSI, MSI, LSI, and VLSI levels. Digital SSI and MSI typically consist of gates and combinations of gates. Design of digital SSI and MSI is presented in Sec. 6.3, and consists largely of the design of standard gates. These standard gates are designed to have large noise margins, large fan out, and large load current capability, in order to maximize their versatility.

In principle, the basic gates are sufficient for the design of any digital integrated circuit, no matter how complex. In practice, modifications are necessary in the basic gates and MSI circuits like flip-flops, registers, adders, etc., when such circuits are to be employed in LSI or VLSI design. For example, circuits to be interconnected on the same chip can be designed with lower noise margins, reduced load driving capability, and smaller logic swing. The resulting benefits are lower power consumption, greater circuit density, and improved reliability. On the other hand, several methodologies have emerged in LSI and VLSI design that are not based on interconnections or modification of SSI and MSI circuits. Both approaches to LSI and VLSI design are presented in the following sections.

© 2006 by Taylor & Francis Group, LLC

6.2 An Overview of the IC Design Process

The effort required for the design of an integrated circuit depends on the complexity of the circuit. The requirement may range from several days effort for a single designer to several months work for a team of designers. **Custom design** of complex integrated circuits is the most demanding. By contrast, **semicustom design** of LSI and VLSI that utilize preexisting designs, such as standard cells and gate arrays, requires less design effort.

IC design is performed at many different levels and Fig. 6.1 is a nonunique depiction of these levels. Level 1 presents the design in terms of subsystems (standard cells, gate arrays, custom subcircuits, etc.) and their interconnections. Design of the system **layout** begins with the floor plan of level 3. It does not involve the layout of individual transistors and devices, but is concerned with the geometric arrangement and interconnection of the subsystems. Level 4 involves the circuit design of the subsystems. Levels 2 and 5 involve system and subcircuit simulations, respectively, which may lead to modifications in levels 1 and/or 4.

Discussion here will focus primarily on the system design of level 1 and the subsystem circuit design of level 4. Lumped under the fabrication process of level 7 are many tasks, such as **mask** generation, process simulation, wafer fabrication, testing, etc. Broadly speaking, floor plan generation is a part of layout. For large ICs, layout design is often relevant to system and circuit design.

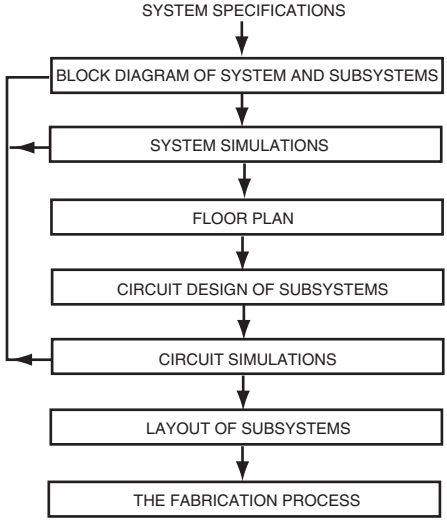


FIGURE 6.1 Different levels in IC design.

6.3 General Considerations in IC Design

Layout or actual arrangement of components on an IC is a design process involving design tradeoffs. Circuit design often influences layout and vice versa, especially in LSI and VLSI. Thus, it is helpful to outline some general considerations in IC design:

- *Circuit components*: Chip area is crucial in IC design. The usual IC components are transistors, resistors, and capacitors. Inductors are uneconomical in area requirements and are hardly ever used in ICs except in certain microwave applications. Transistors require small chip areas and are heavily utilized in ICs. Resistor area requirements increase with resistor values. IC resistors generally range between 50 Ω and 100 kΩ. Capacitors are area intensive and tend to be limited to 100 pF.
- *Isolation regions*: Usually, different components would be placed in different isolation regions. The number of isolation regions, however, should be minimized in the interest of chip area economy by placing more than one component in an isolation region whenever possible. For example, several resistors could share one isolation region.
- *Design rules*: Geometric design rules that specify minimum dimensions for features and separation between features must be followed for a given IC process.
- *Power dissipation*: Chip layout should allow for adequate power dissipation and avoid overheating or development of hot spots on the chip. In low-power circuits such as CMOS ICs, device size is determined by lithographic constraints. In circuits with appreciable power dissipation, device size is determined by thermal constraints and may be much larger than lithographic constraints would allow.

© 2006 by Taylor & Francis Group, LLC

Device size determination from power-density considerations is illustrated with the aid of Fig. 6.2. The device resides close to the surface of the substrate. Thus, heat flow is approximately one-dimensional, from the device to the substrate, although it is actually a three-dimensional flow. Assuming an infinite heat sink of ambient temperature T_A, substrate thickness ΔX, thermal conductivity α, device of surface area A, and temperature $T_A + \Delta T$, the rate of heat flow toward the heat sink, dQ/dt, is given by

$$\frac{dQ}{dt} = \alpha A\left(\frac{\Delta T}{\Delta X}\right) \tag{6.1}$$

The power density or power dissipation per unit area of the device is

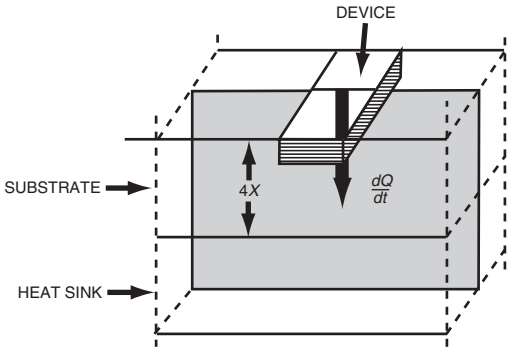


FIGURE 6.2 Heat flow in an IC device.

$$\frac{P}{A} = \alpha\left(\frac{\Delta T}{\Delta X}\right) \tag{6.2}$$

Device Scaling

The trend in IC design, especially VLSI, is progressively toward smaller sizes. Scaling techniques permit the design of circuits that can be shrunk as technological developments allow progressively smaller sizes. Two basic approaches to scaling are full scaling and constant voltage scaling.

- *Full scaling*: All device dimensions, both surface and vertical, and all voltages are reduced by the same scaling factor S.
- *Constant voltage (CV) scaling*: All device dimensions, both surface and vertical, are reduced by the same scaling factor S, but voltages are not scaled. Voltages are maintained at levels compatible with transistor-transistor logic (TTL) supply voltages and logic levels.

Scaling of device dimensions has implications for other device parameters. Full scaling tends to maintain constant electric field strength and, hence, parameters that are no worse off, as device dimensions are reduced, but does not ensure TTL voltage compatibility. Table 6.1 compares effect on device parameters

TABLE 6.1 Effect of Full Scaling and Constant Voltage Scaling on IC Device Parameters

Parameter	Full Scaling	CV Scaling
Channel length, L	$1/S$	$1/S$
Channel width, W	$1/S$	$1/S$
Oxide thickness, t_{OX}	$1/S$	$1/S$
Supply voltage, V_{DD}	$1/S$	1
Threshold voltage, V_{TD}	$1/S$	1
Oxide capacitances, C_{OX}, C_{SW}, C_{FOX}	S	S
Gate capacitance, $C_g = C_{OX}WL$	$1/S$	$1/S$
Transconductances, K_N, K_p	S	S
Current, I_D	$1/S$	$1/S$
Power dissipation per dense, P	$1/S^2$	S
Power dissipation per unit area, P/A	1	S^3
Packing density	S^2	S^2
Propagation delay, t_p	$1/S$	$1/S^2$
Power-delay product, Pt_p	$1/S^3$	$1/S$

© 2006 by Taylor & Francis Group, LLC

of the two scaling approaches. A compromise that is often adopted is to use full scaling for all internal circuits and maintain TTL voltage compatibility at the chip input/output (I/O) pins.

Although many scaling relationships are common to MOS field effect transistors (MOSFETs) and bipolar ICs (Keyes, 1975), the scaling relationships of Table 6.1 more strictly applies to MOSFETs. Bipolar doping levels, unlike MOSFETs, are not limited by oxide breakdown. Thus, in principle, miniaturization can be carried further in bipolar processing technology. However, bipolar scaling is more complex. One reason is that the junction voltages required to turn on a bipolar junction transistor (BJT) does not scale down with dimensions.

Geometric Design Rules

Design rules specify minimum device dimensions, minimum separation between features and maximum misalignment of features on an IC. Such rules tend to be process and equipment dependent. For example, a design rule for a 2-μm process may not be appropriate for a 0.5-μm process. Design rules should protect against fatal errors such as short circuits due to excessive misalignment of features, or open circuits due to too narrow a metal or polysilicon conductive path.

Generalized design rules that are portable between processes and scalable to facilitate adaptation to shrinking minimum geometries as processes evolve are desirable. Other advantages of generalized design rules include increased design efficiency due to fewer levels and fewer rules, automatic translation to final layout, layout-rule and electrical-rule checking, simulation, verification, etc. The Mead-Conway approach (1980) to generalized design rules is to define a scalable and process-dependent parameter, lambda (λ), as the maximum misalignment of a feature from its intended position on a wafer or half the maximum misalignment of two features on different mask layers. Table 6.2 shows a version of the Mead-Conway scalable design rules for NMOS (Fabricius, 1990).

CMOS ICs utilize both NMOS and PMOS devices. Starting with a p-substrate, the NMOS would be fabricated on the p-substrate and the PMOS in an n-well, and vice versa. With the addition of an n-well, p-well or twin tub process, CMOS fabrication is similar to that for NMOS, although more complex. Table 6.3 (Fabricius, 1990) shows the Mead-Conway scalable CMOS design rules. The dimensions are given in multiples of λ and the rules are specified by the MOS Implementation System (MOSIS) of the

TABLE 6.2 MOS Implementation System (MOSIS) NMOS Design Rules

Mask Level	Feature	Size (times λ)
N^+ Diffusion	diffusion width	2
	diffusion spacing	3
Implant mask	implant-gate overlap	2
	implant to gate spacing	1.5
Buried contact mask	contact to active device	2
	overlap with diffusion	1
	contact to poly spacing	2
	contact to diffusion spacing	2
Poly mask	poly width	2
	poly spacing	2
	poly to diffusion spacing	1
	gate extension beyond diffusion	2
	diffusion to poly edge	2
Contact mask	contact width	2
	contact-diffusion overlap	1
	contact-poly overlap	1
	contact to contact spacing	2
	contact to channel	2
	contact-metal overlap	1
Metal mask	metal width	3
	metal spacing	3

© 2006 by Taylor & Francis Group, LLC

TABLE 6.3 MOSIS Portable CMOS Design Rules

Mask Level	Feature	Size (times λ)
n-well and *p*-well	well width	6
	well to well spacing	6
n^+, p^+ active diffusion or implant	active width	3
	active to active spacing	3
	source/drain to well edge	6
	substrate/well contact	3
	active to well edge	3
Poly mask	poly width or spacing	2
	gate overlap of active	2
	active overlap of gate	2
	field poly to active	1
p-select, *n*-select	select-space (overlap) to (of) channel	3
	select-space (overlap) to (of) active	2
	select-space (overlap) to (of) contact	1
Simpler contact to poly	contact size	2 × 2
	active overlap of contact	2
	contact to contact spacing	2
	contact to gate spacing	2
Denser contact to poly	contact size	2 × 2
	poly overlap of contact	1
	contact spacing on same poly	2
	contact spacing on different poly	5
	contact to non-contact poly	4
	space to active short run	2
	space to active long run	3
Simpler contact to active	contact size	2 × 2
	active overlap of contact	2
	contact to contact spacing	2
	contact to gate spacing	2
Denser contact to active	contact size	2 × 2
	active overlap of contact	1
	contact spacing on same active	2
	contact spacing on different active	6
	contact to different active	5
	contact to gate spacing	2
	contact to field poly short run	2
	contact to field poly long run	3
Metal 1	width	3
	metal 1 to metal 1	3
	overlap of contact to poly	1
	overlap of contact to active	1
via	size	2 × 2
	via-to-via separation	2
	metal 1/via overlap	1
	space to poly or active edge	2
	via to contact spacing	2
Metal 2	width	3
	metal 2 to metal 2 spacing	4
	metal overlap of via	1
Overglass	bonding pad (with metal 2 undercut)	100 × 100 μm
	probe pad	75 × 75 μm
	pad to glass edge	6 μm

© 2006 by Taylor & Francis Group, LLC

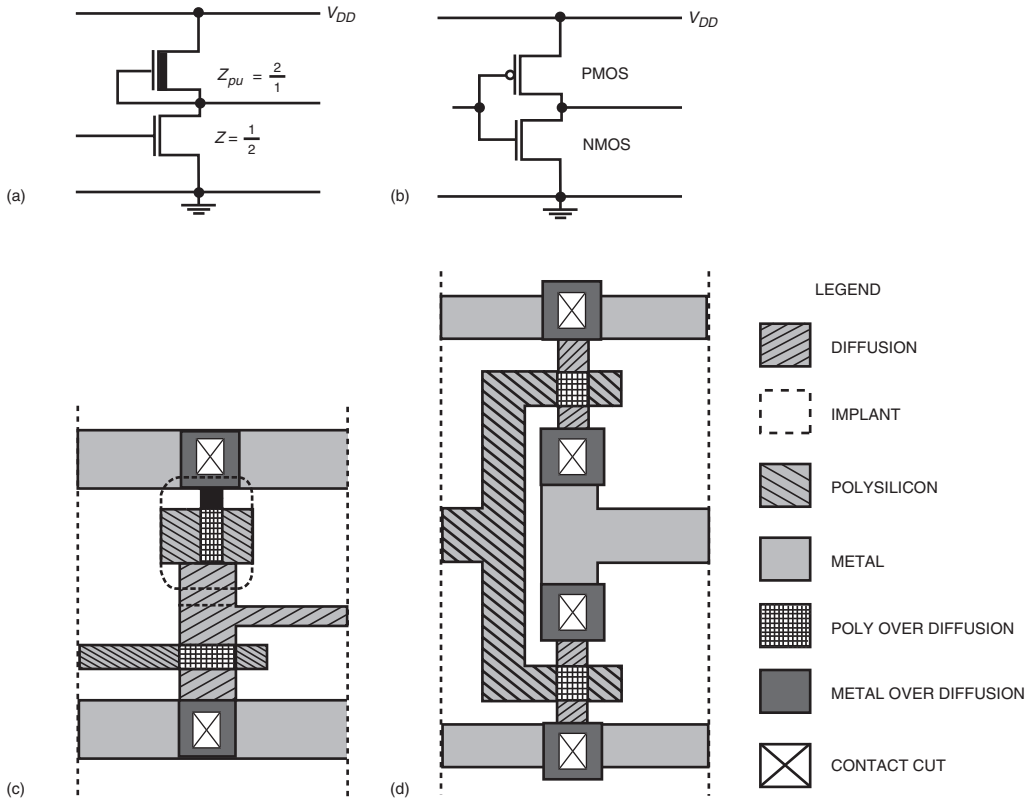


FIGURE 6.3 Layout design for NMOS inverter with depletion load and a CMOS inverter: (a) NMOS inverter, (b) CMOS inverter, (c) NMOS inverter layout, (d) CMOS inverter layout.

University of Southern California Information Sciences Institute. Further clarification of the rules may be obtained from MOSIS manuals (USCISI, 1984, 1988).

Figure 6.3 illustrates the layout of an NMOS inverter with depletion load and a CMOS inverter, employing the NMOS and CMOS scalable design rules just discussed. Figure 6.3(a) and Figure 6.3(b) show the circuit diagrams for the NMOS and CMOS, respectively, with the aspect ratios Z defined as the ratio of the length to the width of a transistor gate. Figure 6.3(c) and Figure 6.3(d) give the layouts for the NMOS and CMOS, respectively.

A simplified design rule for a bipolar *npn* comparable to those previously discussed for NMOS and CMOS is presented in Table 6.4. As before, the minimum size of a feature in the layout is denoted by lambda (λ). The following six masks are required for the fabrication: n^+ buried-layer diffusion, p^+ isolation diffusion, p base region diffusion, n^+ emitter and collector diffusions, contact windows, and metalization.

6.4 Design of Small-Scale and Medium-Scale Integrated Circuits

Gates are the basic building blocks in digital integrated circuits. Small-scale integrated circuits are essentially gate circuits, and medium-scale integrated circuits are circuits employing several gates. Gates, in turn, are based on inverters and can be realized from inverter circuits with some modifications, especially those modifications that allow for multiple inputs. This section will start with a discussion of inverters and gates.

© 2006 by Taylor & Francis Group, LLC

TABLE 6.4 Simplified Design Rules for *npn* Transistor

Mask Level	Feature	Size (times λ)
Isolation mask	width of isolation wall	1
	wall edge to buried layer spacing	2.5
Base mask	base to isolation region spacing	2.5
Emitter	area	2 × 2
	emitter to base diffusion spacing	1
	multiple emitter spacing	
Collector contact	area	1 × 5
	n^+ to base diffusion	1
Contact windows	base contact	1 × 2
	emitter contact	1 × 1
	collector contact	1 × 2
	base contact to emitter spacing	1
Metallization	width	1.5
	metal to metal spacing	1

NMOS Inverters

A resistive load NMOS inverter, its output characteristics, and its voltage transfer characteristic are shown in Fig. 6.4. The loadline is also shown on the output characteristics. Resistive load inverters are not widely used in ICs because resistors require long strips and, hence, large areas on the chip. A solution to this problem is to use active loads, since transistors are economical in chip area.

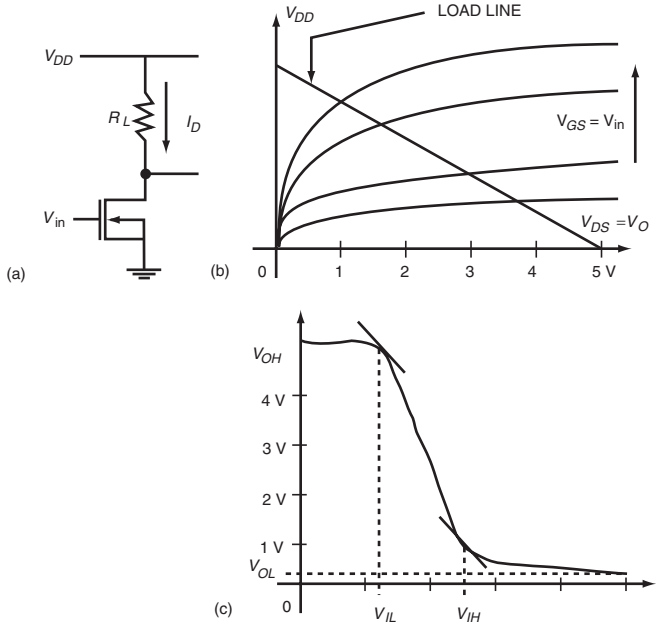


FIGURE 6.4 NMOS resistive load inverter: (a) resistive load NMOS inverter, (b) output characteristics, (c) transfer characteristic.

© 2006 by Taylor & Francis Group, LLC

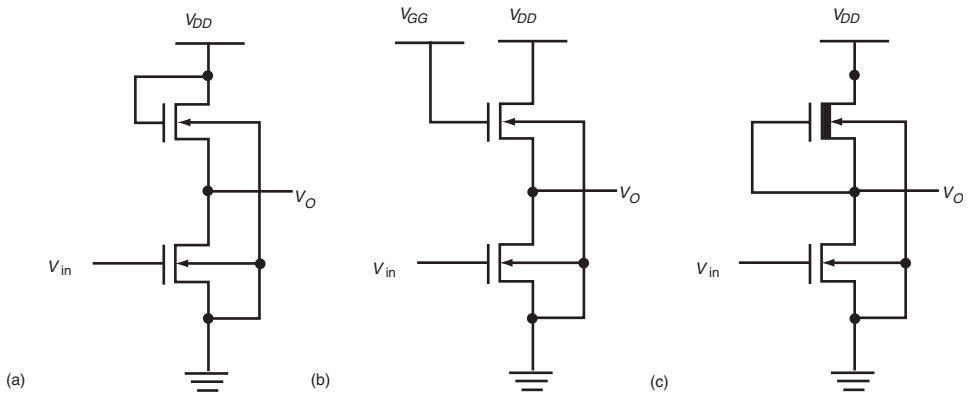


FIGURE 6.5 NMOS inverters with different types of active loads: (a) saturated enhancement load, (b) linear enhancement load, (c) depletion load.

Figure 6.5 shows three NMOS inverters with three types of NMOS active loads: saturated enhancement load, linear enhancement load, and depletion load. One basis for comparison between these inverters is the geometric ratio K_R, which is defined as $Z_{\text{pu}}/Z_{\text{pd}}$. Z denotes the ratio of length to width of a transistor channel. The subscript pu refers to the pull-up or load device, whereas the subscript pd refers to the pull-down or driving transistor.

The saturated enhancement load inverter overcomes much of the area disadvantage of the resistive load inverter. When carrying the same current and having the same pull-down transistor as the resistive inverter, however, K_R is large for the saturated enhancement load inverter, indicating load transistor area minimization is still possible. This configuration yields a smaller logic swing relative to the resistive load inverter, however, because the load transistor stops conducting when its $V_{GS} = V_{DS}$ decreases to V_T. Thus, for this inverter, $V_{OH} = V_{DD} - V_T$.

In Fig. 6.5(b), because V_{GG} is greater than $V_{DD} + V_T$, V_{DS} is always smaller than $V_{GS} - V_T$; thus, the load always operates in the linear region. This results in a linear enhancement load NMOS inverter. The high value of V_{GG} also ensures that V_{GS} is always greater than V_T, so that the load remains on and V_{OH} pulls up to V_{DD}. The linear enhancement load configuration, however, requires a load transistor of larger area relative to the saturated enhancement load inverter, and requires additional chip area for the V_{GG} contact.

In the depletion NMOS load inverter of Fig. 6.5(c), $V_{GS} = 0$, thus the load device is always on and V_{OH} pulls all the way to V_{DD}. This configuration overcomes the area disadvantage without incurring a voltage swing penalty. It is, therefore, the preferred alternative. The performance of the NMOS inverters with the four different types of loads are graphically compared in Fig. 6.6(a) and Fig. 6.6(b). Both the loadlines and the voltage transfer characteristics were obtained from SPICE simulation. Figure 6.6(a) shows the loadlines superimposed on the output characteristics of the pull-down transistor, which is the same for the four inverters. R_L is 100 kΩ and each inverter has $V_{DD} = 5$ V, $V_{OL} = 0.2$ V and $I_{D\,\max} = 48\ \mu$A. Note that V_{OH} falls short of V_{DD} for the saturated enhancement load inverter but not for the others. Figure 6.6(b) shows the voltage transfer characteristics (VTC) for the four inverters. V_{OH} is again shown to be less than V_{DD} for the saturated enhancement load. Note, also, that the depletion load VTC more closely approaches the ideal inverter VTC than any of the others.

The loadlines of Fig. 6.5(a) are easy to generate. Consider, for example the depletion NMOS load. V_{GS} is fixed at 0 V, so that its output characteristic consists of only the curve for $V_{GS} = 0$. I_D is always the same for the load and driving transistor, but their V_{DS} add up to V_{DD}. Thus, when V_{DS} is high for one transistor, it is low for the other. The loadline is obtained by shifting the origin for V_{DS} for the load characteristic to V_{DD}, reflecting it about the vertical axis through V_{DD} and superimposing it on the V–I characteristics for the driving inverter.

© 2006 by Taylor & Francis Group, LLC

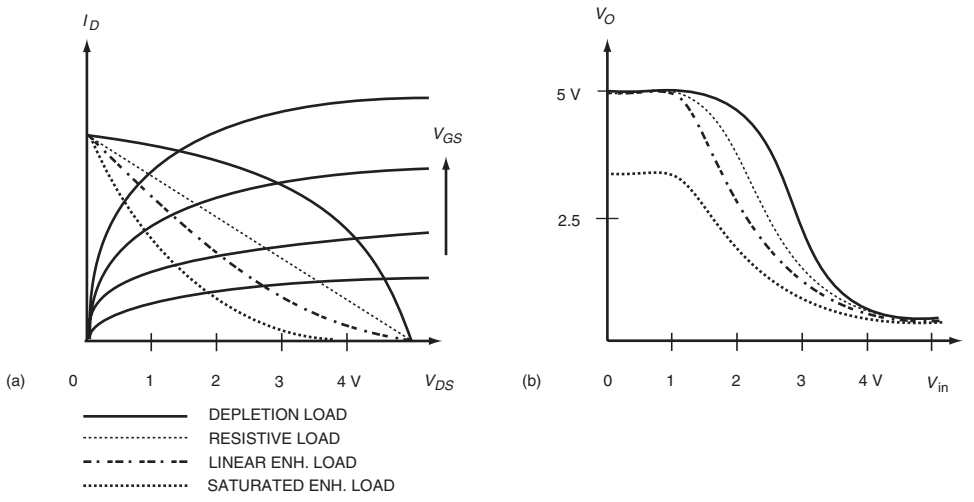


FIGURE 6.6 Performance of NMOS inverters with different types of loads: (a) output characteristics and load lines, (b) voltage transfer characteristics.

The voltage transfer characteristics are best generated by computer simulation. Useful insights, however, can be gained from an analysis yielding the critical voltages V_{OH}, V_{OL}, V_{IH}, V_{IL}, and V_O for any specified V_{in}. The NMOS currents hold the key to such an analysis. Threshold voltages are different for enhancement and depletion NMOS transistors, but the drain current equations are the same. The drain current is given in the linear region and the saturated region, respectively, by

$$I_D = K_n\left[2(V_{GS} - V_T)V_{DS} - V_{DS}^2\right]; \quad V_{DS} \le V_{GS} - V_T \tag{6.3}$$

$$I_D = K_n(V_{GS} - V_T)^2; \quad V_{DS} \ge V_{GS} - V_T \tag{6.4}$$

where

V_T = threshold voltage
$K_n = \frac{\mu_n C_{\text{ox}}}{2}\left(\frac{w}{L}\right)$ = transconductance
μ_n = electron channel mobility
C_{ox} = gate capacitance per unit area

Similar definitions apply to PMOS transistors.

Consider the VTC of Fig. 6.7(a) for a depletion load NMOS inverter. For the region $0 < V_{\text{in}} < V_T$ the driving transistor is off, so $V_{OH} = V_{DD}$. At A, V_{in} is small; thus, for the driving transistor, $V_{DS} = V_O > V_{\text{in}} - V_T = V_{GS} - V_T$. For the load $V_{DS} = V_{DD} - V_O$ is small. Hence, the driver is saturated and load is linear. Similar considerations lead to the conclusions as to the region in which each device operates, as noted in the figure. To find V_{IL} and V_{IH}, the drain currents for the appropriate region of operation for points A and C, respectively, are equated, for both transistors. Differentiating the resulting equations with respect to V_{in} and applying the condition that $dV_O/dV_{\text{in}} = -1$ yields the required critical voltages. Equating drain currents for saturated load and linear driver at $V_{\text{in}} = V_{DD}$ and solving yields V_{OL}. The output voltage V_O may be found at any value of V_{in} by equating the drain currents for the two transistors, appropriate for that region of V_{in}, and solving for V_O at the given V_{in}.

NMOS Gates

Only NOR and NAND need be considered because these are more economical in chip area than OR and AND, and any logic system can be implemented entirely with either NOR or NAND. By connecting driving

© 2006 by Taylor & Francis Group, LLC

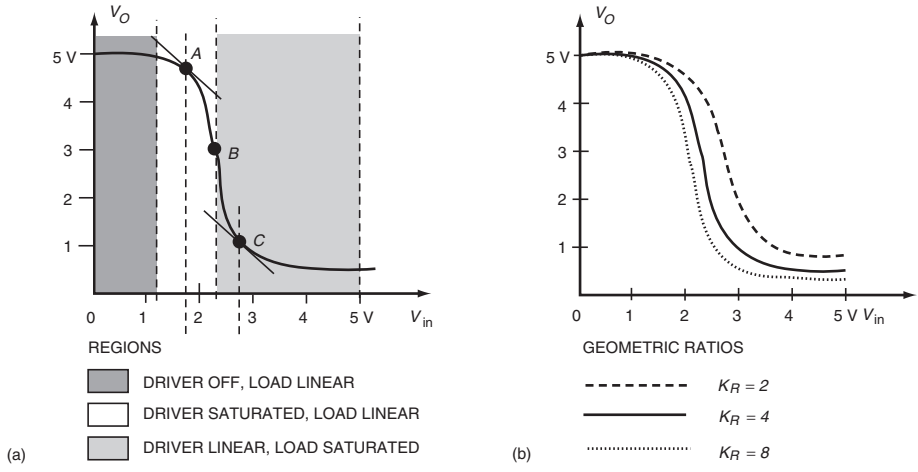


FIGURE 6.7 Regions of the VTC and VTCs for different geometric ratios for the depletion load NMOS inverters: (a) depletion load NMOS inverter VTC and its regions, (b) VTCs for different geometric ratios.

transistors in parallel to provide the multiple inputs, the NMOS inverter is easily converted to a NOR gate as shown in Fig. 6.8(a). By connecting driving transistors in series as in Fig. 6.8(b), an NAND gate is obtained.

The NMOS, NOR, and NAND gates are essentially modifications of the depletion load NMOS inverter of Fig. 6.5(c). They all have the same depletion load, and their performance should be similar. For the same value of V_{DD}, each has $V_{OH} = V_{DD}$, and they should have the same V_{OL} and the same drain current when $V_O = V_{OL}$. In Fig. 6.5(c), the depletion load inverter has $Z_{\text{pu}} = 2$, $Z_{\text{pd}} = \frac{1}{2}$ and $K_R = 4$. Thus, Z_{pu} is 2 for the NOR and the NAND gate. With only one driving transistor on in the NOR gate, the drain current should be sufficient to ensure that $V_O = V_{OL}$. Thus, for each of the driving transistors, $Z_I = \frac{1}{2}$, as for the depletion load inverter. For the NAND gate, the equivalent series combination of Z_{pd} (proportional to drain-source resistance) should also be $\frac{1}{2}$ to allow the same value of V_O, leading to $Z_I = \frac{1}{4}$ for each driving transistor in the NAND gate. Thus, K_R is 4 for the inverter, 4 for the NOR gate, and 8 for the NAND. As the number of inputs increases, K_R increases for the NAND but not for the NOR. It is clear that NMOS

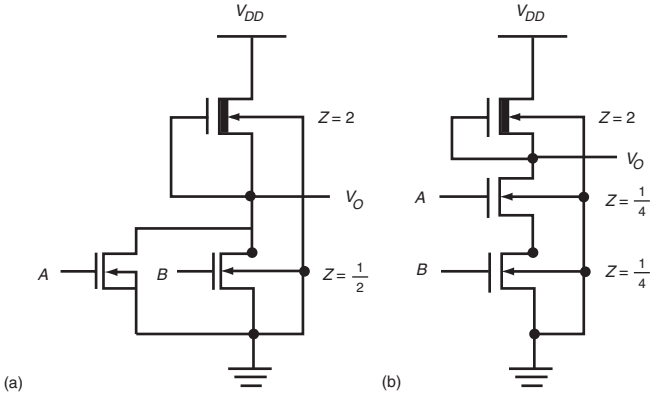


FIGURE 6.8 NMOS gates: (a) NMOS NOR gate, (b) NMOS NAND gate.

© 2006 by Taylor & Francis Group, LLC

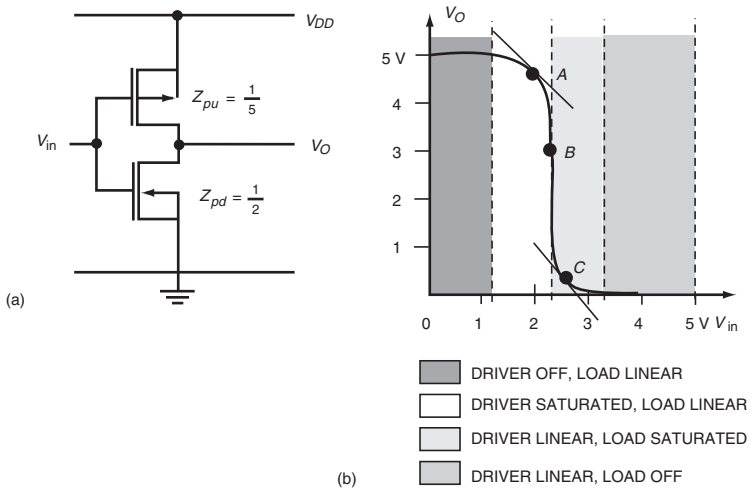


FIGURE 6.9 The CMOS inverter and its voltage transfer characteristics: (a) CMOS inverter, (b) CMOS transfer characteristic and its regions.

NAND gates are wasteful of chip area relative to NMOS NOR. Hence, NOR gates are preferred (and the NOR is the standard gate) in NMOS.

CMOS Inverters

As shown in Fig. 6.9(a), the CMOS inverter consists of an enhancement NMOS as the driving transistor, and a complementary enhancement PMOS load transistor. The driving transistor is off when V_{in} is low, and the load transistor is off when V_{in} is high. Thus, one of the two series transistors is always off (equivalently, drain current and power dissipation are zero) except during switching, when both transistors are momentarily on. The resulting low-power dissipation is an important CMOS advantage and makes it an attractive alternative in VLSI design.

NMOS circuits are ratioed in the sense that the pull up never turns off, and V_{OL} is determined by the inverter ratio. CMOS is ratioless in this sense, since V_{OL} is always the negative rail. If one desires equal sourcing and sinking currents, however, the pull-up device must be wider than the pull-down device by the ratio of the electron-to-hole mobilities, typically about 2.5 to 1. This also gives a symmetrical voltage transfer curve, with the voltage at which $V_{\text{in}} = V_O$ having a value of $V_{DD}/2$. This voltage is referred to as the inverter voltage V_{inv}.

The voltage transfer for the CMOS inverter is shown in Fig. 6.9(b). Note that the voltage transfer characteristic approaches that of the ideal logic inverter. These characteristics are best obtained with computer circuit simulation programs. As with the depletion load NMOS inverter, useful insights may be gained by performing an analytical solution. The analysis proceeds as previously described for the depletion load NMOS inverter. Note that the VTC of Fig. 6.9(b) has been divided into regions as in Fig. 6.7(a). In each region, the appropriate expressions for the load and driving transistor drain currents are equated so that V_O can be computed for any given V_{in}. To find V_{IL} and V_{IH}, the condition that $dV_O/dV_{\text{in}} = -1$ at such critical voltages is applied to the drain current equation. Note that the drain current equations for the PMOS are the same as for NMOS (Eqs. 6.3 and 6.4), except for reverse voltage polarities for the PMOS.

CMOS Gates

CMOS gates are based on simple modifications to the CMOS inverter. Figure 6.10(a) and Figure 6.10(b) show that the CMOS NOR and NAND gates are essentially CMOS inverters in which the load and driving

© 2006 by Taylor & Francis Group, LLC

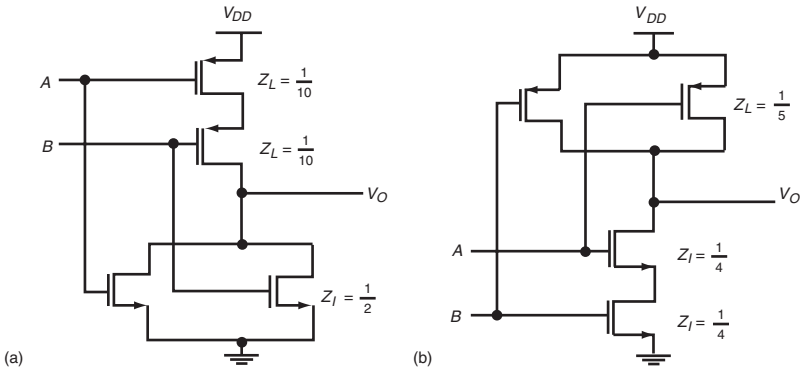


FIGURE 6.10 CMOS gates: (a) CMOS NOR gate, (b) CMOS NAND gate.

transistor are replaced by series or parallel combinations (as appropriate) of PMOS and NMOS transistors, respectively.

Suppose the NOR gate of Fig. 6.10(a) is to have the same V_{DD} and V_{inv} as the CMOS inverter of Fig. 6.9(a), then the equivalent Z_{pu} and Z_{pd} for the NOR gate should equal those for the inverter. Since only one of the parallel pull-down transistors needs be on in the NOR to ensure $V_O = 0$ V, $Z_I = Z_{\text{pd}} = \frac{1}{2}$, as for the inverter. For the series load, however, $Z_L = \frac{1}{10}$ to give equivalent $Z_{\text{pu}} = \frac{1}{5}$. If the NAND gate of Fig. 6.10(b) is to have the same V_{inv} as the said inverter, similar arguments lead to $Z_I = \frac{1}{4}$ and $Z_L = \frac{1}{5}$ for the NAND. Thus, $K_R = 0.4$ for the inverter, 0.2 for the NOR, and 0.8 (closer to unity) for NAND. Hence, NAND is the standard gate in CMOS. Another way of putting this is that for the given Z values, if the channel length L is constant, then the widths of the loads for the inverter, NOR, and NAND are in the ratio 1:2:1. Thus, the NOR requires more chip area, and this larger area requirement increases with the number of inputs.

Bipolar Gates

The major bipolar digital logic families are TTL, emitter-coupled logic (ECL), and integrated injection logic (I^2L). Within each logic family, there are subclassifications, for example, the Schottky transistor logic (STL), and the integrated Schottky logic (ISL), which developed from the basic I^2L. Bipolar gates have faster switching speeds but greater power dissipation than CMOS gates. The most popular bipolar gate in SSI is the low-power Schottky TTL, which has moderate power dissipation and propagation delay. The fastest switching bipolar family is the ECL, but it has relatively high-power dissipation. The highest packing density is achieved with I^2L and its relatives with low-power dissipation and moderate switching speeds. A better comparison of logic families should be based on the power-delay product, which takes into account both power dissipation and propagation delay.

Medium-Scale Integrated Circuits

MSI circuits have between 10 and 100 transistors per chip. They are built from inverters and basic logic gates with hardly any modifications. They require minimal design effort beyond putting together and interconnecting logic gates. Examples of MSI circuits are flip-flops, counters, registers, adders, multiplexers, demultiplexers, etc.

6.5 LSI and VLSI Circuit Design

Semicustom design is a heavily utilized technique in LSI and VLSI design. In this technique, largely predesigned subcircuits or cells are interconnected to form the desired, larger circuit. Such subcircuits are usually highly regular in nature, so that the technique leads to highly regular circuits and layouts.

© 2006 by Taylor & Francis Group, LLC

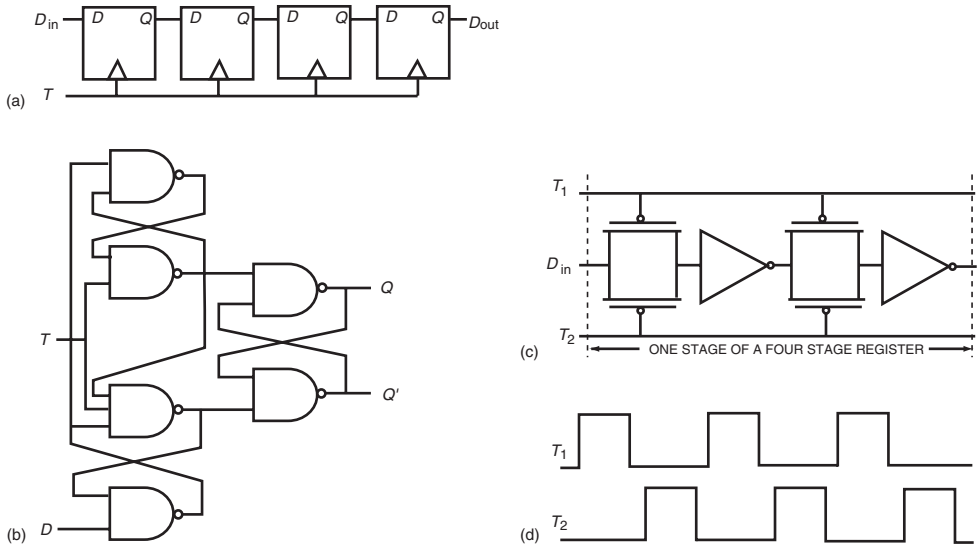


FIGURE 6.11 Conventional (static) and dynamic 4-b shift register: (a) conventional static shift register, (b) Dflip-flop, (c) dynamic shift register, (d) two-phase clock pulses.

Multiphase Clocking

Multiphase clocking is an important technique that can be used to reduce device count in LSI and VLSI circuits. To illustrate the savings that can be realized with the technique, device count is compared for a conventional design of a 4-b shift register employing *D* flip-flops based on CMOS NAND gates and a 4-b shift register employing two-phase clocks and CMOS technology.

Both designs are shown in Fig. 6.11. Figure 6.11(a) shows the conventional design for the shift register, which employs a single phase clock signal, whereas Fig. 6.11(b) shows the circuit realization of each D flip-flop with CMOS NAND gates (Taub and Schilling, 1977). The device count for each in this design is obtained as follows:

- 5 two-input CMOS NAND gates, 4 transistors each: 20 transistors
- 1 three-input CMOS NAND gate: 6 transistors
- Number of transistors per *D* flip-flop: 26
- Total number of transistors for the 4-b register: 104

The second design, which employs two-phase clocking, is shown in 6.11(c), whereas the nonoverlapping clock signals are shown in Fig. 6.11(d). Note that each flip-flop now consists of two CMOS transmission gates and two CMOS inverters. Thus, there are 8 transmission gates and 8 inverters in the 4-b shift register. Device count for this design is as follows:

- Number of transistors for 8 transmission gates: 16
- Number of transistors for 8 CMOS inverters: 16
- Total number of transistors for the 4-b register: 32

In the preceding example, employing two-phase clocking helped to reduce device count to less than one-third of the requirement in the conventional static design. This gain, however, is partly offset by the need for more complex clocking and the fact that the shift register is now dynamic. To avoid loss of data

© 2006 by Taylor & Francis Group, LLC

due to leakage through off transistors, the clock must run above a minimum frequency. The times required to charge and discharge capacitive loads determine the upper clock frequency.

6.6 Increasing Packing Density and Reducing Power Dissipation in MOS Circuits

CMOS gates have much lower power dissipation than NMOS gates. This is a great advantage in LSI and VLSI design. Standard CMOS gates, however, require two transistors per input and, therefore, have higher device count than NMOS gates that require one driving transistor per input, plus one depletion load transistor, irrespective of the number of inputs (Mavor, Jack, and Denyer, 1983). This NMOS feature is put to advantage in applications such as semiconductor memories and programmable logic arrays, which will be discussed later. In addition to requiring a higher device count, it is necessary to isolate the PMOS and NMOS transistors in the CMOS and to employ metal interconnection between their drains, which are of opposite conductivity. Consequently, gate count per chip for NMOS is about half that of CMOS, using the same design rules.

Figure 6.12 shows a CMOS domino logic circuit in which clocking is employed in an unconventional CMOS circuit to provide both high density and low-power dissipation. When T is low, Q_1 is off, so there is no path to ground irrespective of the logic levels at the inputs A, B, C, D and E. Q_2 is on, so that the parasitic capacitance C_1 charges to V_{DD}. When T is high, Q_2 is off and Q_1 is on. Thus if both A and B, or both C and D, or all of A, B, C, and D are high, a path exists from C_1 to ground, and it discharges. Otherwise, C_1 remains high (but for slow leakage), and the valid logic $(A\ B) + (C + D)$ appears at the output F. Note that this circuit has only two load PMOS transistors, and only one driving transistor is required for each additional input. Thus, device count is minimized by using complex instead of simple logic functions. Each transistor, except those in the output inverter, may be minimum size, since they

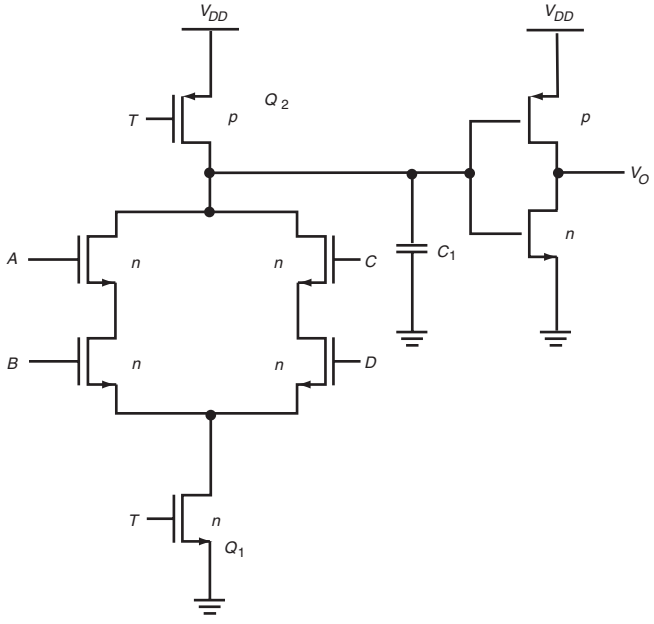


FIGURE 6.12 CMOS domino AND-OR logic.

© 2006 by Taylor & Francis Group, LLC

are required only to charge or discharge C_1. Power dissipation is low as for standard CMOS, because no steady-state current flows.

6.7 Gate Arrays

Gate arrays are a category of semicustom integrated circuits typically containing 100 to several thousand gate cells arranged in rows and columns on a chip. The gate cell may be a NAND, NOR, or other gate. Often, each gate cell is a set of components that could be interconnected to form the desired gate or gates. Identical gate cell pattern is employed, irrespective of chip function. Consequently, gate arrays can be largely processed in advance (Reinhard, 1987). Less effort is required for design with gate arrays since only the masks needed for interconnection are required to customize a chip for a particular application.

Figure 6.13 illustrates a gate array in various levels of detail and possible interconnections within a cell. The floor plan of Fig. 6.13(a) shows that there are 10 columns of cells with 10 cells per column, for a total of 100 cells in the chip. The cell layout of Fig. 6.13(b) shows that there are 4 NMOS and 4 PMOS transistors per cell. Thus there are a total of 800 transistors in the chip. The transistor channels are under the polysilicon and inside the diffusion areas. Figure 6.13(c) shows the cell layout with interconnection to form an NAND gate, whereas Fig. 6.13(d) shows the circuit equivalent of a cell.

Because of their simplicity, a significant amount of wiring is required for interconnections in gate arrays. Good computer software is essential for designing interconnections. In practice, wiring channels tend to fill up, so that it is difficult to utilize more than 70% of the cells on a chip (Alexander, 1985). The standard

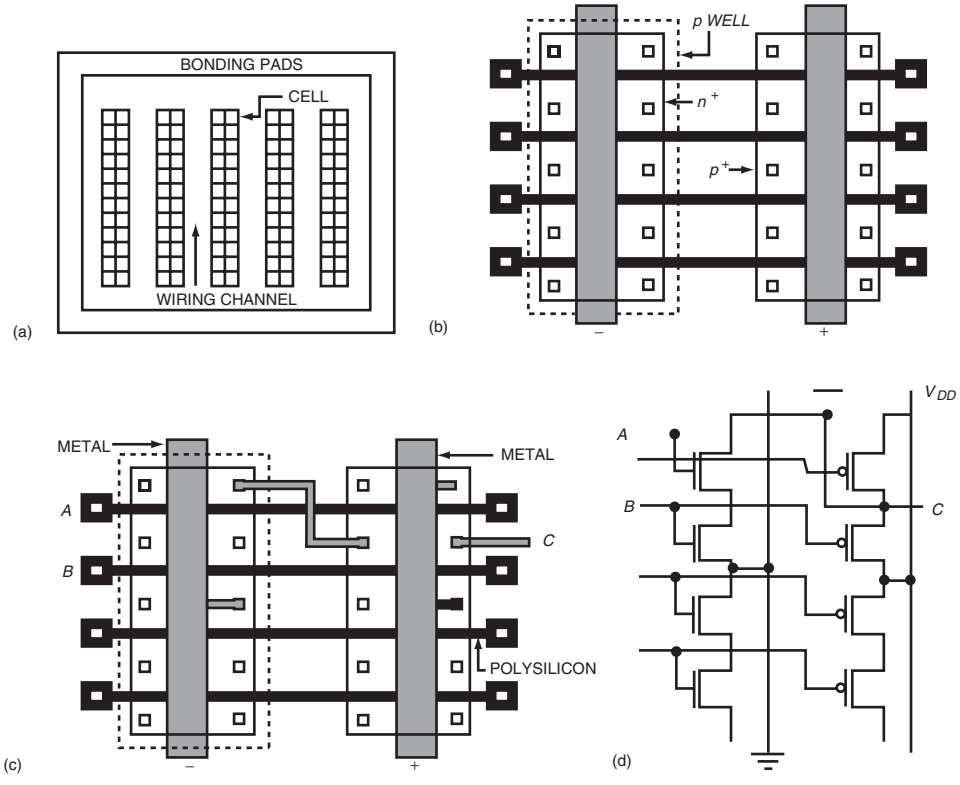


FIGURE 6.13 Gate array at various levels of detail: (a) cell structure, (b) transistor structure, (c) NAND gate interconnection, (d) equivalent circuit.

© 2006 by Taylor & Francis Group, LLC

cell approach discussed next reduces this problem, to some extent, by permitting use of more complex logic functions or cells.

6.8 Standard Cells

In the **standard cell** approach, the IC designer selects from a library of predefined logic circuits or cells to build the desired circuit. In addition to the basic gates, the **cell library** usually includes more complex logic circuits such as exclusive-OR, AND-OR-INVERT, flip-flops, adders, read only memory (ROM), etc.

The standard cell approach to design is well suited to automated layout. The process consists of selecting cells from the library in accordance with the desired circuit functions, the relative placement of the cells, and their interconnections. The floor plan for a chip designed by this method is similar to the floor plan for a gate array chip as shown in Fig. 6.13(a). Note, however, that the designer has control over the number and width of wiring channels in this case. Layout for a cell is always the same each time the cell is used, but the cells used and their relative placement is unique to a chip. Thus, every mask level is unique in this approach, and fabrication is more involved and more costly than in the gate array approach (Hodges and Jackson, 1988).

6.9 Programmable Logic Devices

Programmable logic devices (PLDs) are a class of circuits widely used in LSI and VLSI design to implement two-level, sum-of-products, boolean functions. Multilevel logic can be realized with Weinberger arrays or gate matrices (Fabricius, 1990; Weinberger, 1967). Included among PLDs are programmable logic arrays (PLAs), programmable array logic (PAL), and ROM. The AND-OR structure of the PLA, which can be used to implement any two-level function, is the core of all PLDs. The AND-OR function is often implemented with NOR-NOR or NAND-NAND logic.

PLDs have the advantage of leading to highly regular layout structure. The PLD consists of an AND plane followed by an OR plane. The logic function is determined by the presence or absence of contacts or connections at row and column intersections in a single conducting layer. Programming or establishment of appropriate contacts may be accomplished during fabrication. Alternatively, the PLDs may be user programmable by means of fuse links.

Figure 6.14 shows the three types of PLDs. Hollow diamonds at row/column intersections in an AND or OR plane indicates that the plane is programmable. Presence of solid diamonds in some row/column intersections indicate that the logic for that plane is already defined and fixed. The PLD is a PLA if both the AND and OR planes are programmable, a PAL if only the AND plane is programmable, and a ROM if only the OR plane (the decoder in this case) is programmable. Because PLAs are programmable in both planes, they permit more versatile logic realizations than PALs. Also, the PAL can be considered a special case of the PLA. Thus, only the PLA is discussed further.

Programmable Logic Array

PLAs provide an alternative to implementation of combinational logic that results in highly regular layout structures. Consider, for example, a PLA implementation of the following sum of product expressions:

$$Y_1 = \bar{I}_0 \bar{I}_1 + I_0 \bar{I}_2 \tag{6.5}$$

$$Y_2 = \bar{I}_0 I_1 I_2 + I_0 \bar{I}_2 \tag{6.6}$$

$$Y_3 = \bar{I}_0 \bar{I}_1 \bar{I}_2 + \bar{I}_0 I_1 I_2 \tag{6.7}$$

© 2006 by Taylor & Francis Group, LLC

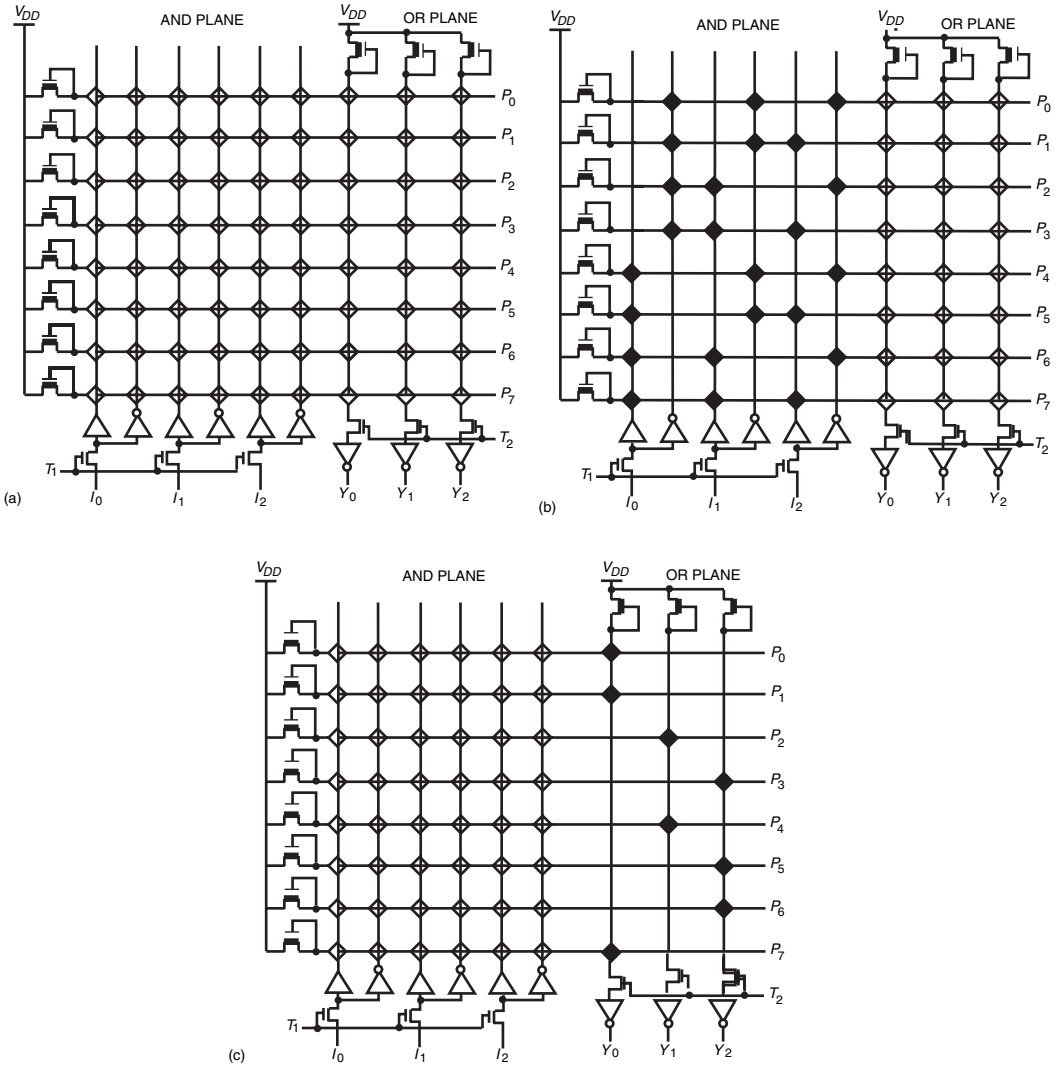


FIGURE 6.14 Types of programmable logic devices: (a) programmable logic array, (b) programmable read only memory, (c) programmable array logic.

The PLA has three inputs and three outputs. In terms of the AND and OR planes, the outputs of the AND plane are

$$P_1 = \bar{I}_0 \bar{I}_2 \tag{6.8}$$

$$P_2 = \bar{I}_0 \bar{I}_1 \bar{I}_2 \tag{6.9}$$

$$P_3 = \bar{I}_0 I_1 I_2 \tag{6.10}$$

$$P_4 = I_0 \bar{I}_2 \tag{6.11}$$

The overall output is the output of the OR plane and can be written in terms of the AND plane outputs as

$$Y_1 = P_1 + P_4 \tag{6.12}$$

$$Y_2 = P_3 + P_4 \tag{6.13}$$

$$Y_3 = P_2 + P_3 \tag{6.14}$$

© 2006 by Taylor & Francis Group, LLC

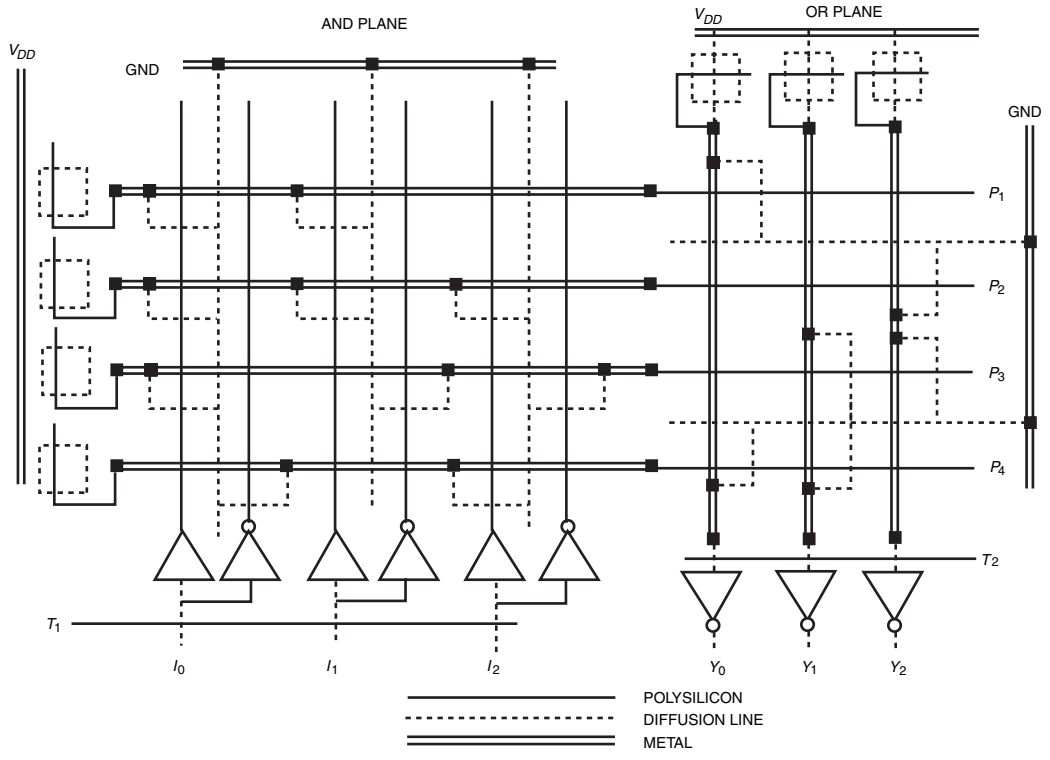


FIGURE 6.15 An NMOS PLA.

Figure 6.15 shows the logic circuit consisting of the AND and the OR planes. Note that each product line in the AND plane is an NMOS NOR gate with one depletion load; the gate of each driving transistor is controlled by an input line. Likewise, each output line in the OR plane is an NMOS NOR gate with driving transistors whose gates are controlled by the product lines. Thus, the PLA employs a NOR–NOR implementation.

The personality matrix for a PLA (Lighthart, Aarts, and Beenker, 1986) gives a good description of the PLA and how it is to be programmed. The personality matrix Q of the PLA of Fig. 6.15 is given by Eq. (6.15).

$$Q = \begin{bmatrix} 0 & 0 & x & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & x & 0 & 1 & 1 & 0 \end{bmatrix} \tag{6.15}$$

The first three columns comprise the AND plane of the matrix, whereas the last three columns comprise the OR plane of the three-input, three-output PLA. In the AND plane, element $q_{ij} = 0$ if a transistor is to link the product line P_i to the input line I_i; $q_{ij} = 1$ if a transistor is to link P_i to $\bar{I}_i$, and q_{ij} is a *don't care* if neither input is to be connected to P_i. In the OR plane, $q_{ij} = 1$ if product line P_i is connected to output Y_j and 0 otherwise.

Figure 6.16 shows the stick diagram layout of the PLA circuit of Fig. 6.15, and illustrates how the regular structure of the PLA facilitates its layout. The input lines to each plane are polysilicon, the output lines from each plane are metal, and the sources of driving transistors are connected to ground by diffused lines. The connecting transistors are formed by grounded, crossing diffusion lines.

© 2006 by Taylor & Francis Group, LLC

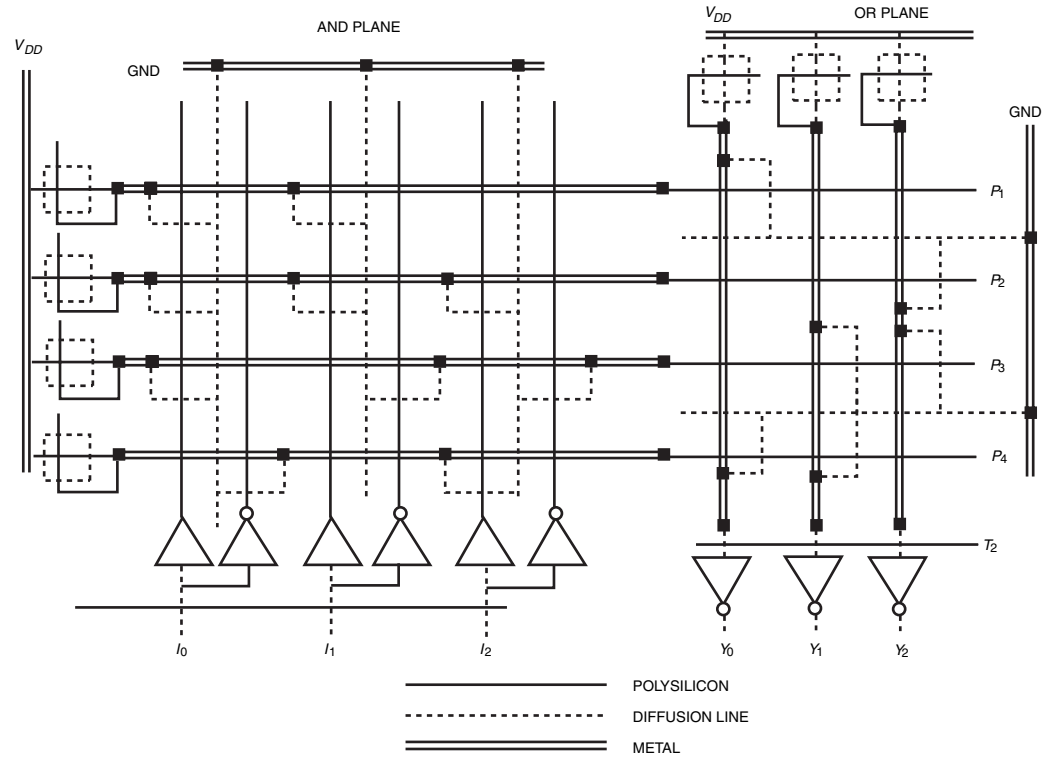


FIGURE 6.16 Stick diagram layout of the PLA shown in Fig. 6.16.

6.10 Reducing Propagation Delays

Large capacitive loads are encountered in many ways in large integrated circuits. Bonding pads are required for interfacing the chip with other circuits, whereas probe pads are often required for testing. Both present large capacitive loads to their drivers. Interconnections within the chip are by means of metal or polysilicon lines. When long, such lines present long capacitive loads to their drivers. Although regular array structures, such as those of gate arrays, standard cells, and PLAs, are very convenient for semicustom design of LSI and VLSI, they have an inherent drawback with regard to propagation delay. Their row and column lines contact many devices and, hence, are very capacitive. The total delay of a long line may be reduced by inserting buffers along the line to restore the signal. Superbuffers are used for interfacing between small gates internal to the chip and large pad drivers and for driving highly capacitive lines.

Resistance-Capacitance (RC) Delay Lines

A long polysilicon line can be modeled as a lumped RC transmission line as shown in Fig. 6.17. Let Δx represent the length of a section of resistance R and capacitance C, and Δt be the time required for the signal to propagate along the section. Let $\Delta V = (V_{n-1} - V_n)\Delta x$, where V_n is the voltage at node n. The difference equation governing signal propagation along the line is (Fabricius, 1990)

$$RC\frac{\Delta V}{\Delta t} = \frac{\Delta^2 V}{\Delta x^2} \tag{6.16}$$

© 2006 by Taylor & Francis Group, LLC

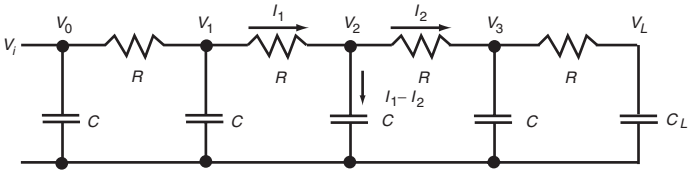


FIGURE 6.17 Lumped circuit model of a long polysilicon line.

As the number of sections becomes very large, the difference equation can be approximated with the differential equation

$$RC\frac{dV}{dt} = \frac{d^2V}{dx^2} \tag{6.17}$$

For a delay line with N sections, matched load $C_L = C$, resistance R and capacitance C per unit length, and overall length L in micrometer, Horowitz (1983) showed that the propagation delay is

$$t_d = \frac{0.7N(n+1)L^2RC}{2N^2} \tag{6.18}$$

Note that the propagation delay is proportional to the square of the length of the line. As N tends to infinity, the propagation delay becomes

$$t_d = \frac{0.7RCL^2}{2} \tag{6.19}$$

To reduce the total delay restoring inverters can be inserted along a long line. Consider as an example, a 5-mm-long polysilicon line with $r = 20\ \Omega/\mu$m and $C = 0.2$ fF/μm. It is desired to find the respective propagation delays, if the number of inverters inserted in the line varies from zero to four. The delay of each inverter is proportional to the length of the segment it drives and is given by $t_I = 0.4$ ns when it is driving a 1-mm-long segment. In each case, the inverters used are spaced uniformly along the line. Let

K = number of inverters used
$K + 1$ = number of sections
ℓ = length per section, $\frac{5\text{mm}}{k+1}$
t_d = total delay

then

$$t_d = \left[(k+1)\left(\frac{0.7RC\ell^2}{2}\right) + 0.4\ell K\right] ns \tag{6.20}$$

From Eq. (6.20), the propagation delays can be calculated. The delay for each number of inverters as a percentage of the unbuffered line delay is also computed. The results are tabulated in Table 6.5.

The results in the table show that the propagation delay decreases as the number of inverters is increased. The improvement in propagation delay, however, is less dramatic for each additional inverter than the one preceding it. The designer would stop increasing the number of inverters when the incremental gain no longer justifies an additional inverter. If the number of inverters is even, there is no inversion of the overall signal.

Superbuffers

Propagation delays can be reduced without excessive power consumption by using superbuffers. These are inverting or noninverting circuits that can source and sink larger currents and drive large capacitive

© 2006 by Taylor & Francis Group, LLC

TABLE 6.5 Improvement in Propagation Delay with Increase in Number of Line Buffered

No. of Inverters K	Total Delay t_d, ns	% of Unbuffered Line Delay
0 (unbuffered)	35.0	100
1	18.5	52.86
2	13.0	37.14
3	10.25	29.29
4	8.6	24.57

loads faster than standard inverters. Unlike ratioed NMOS inverters in which the pull-up current drive capability is much less than the pull-down capability, superbuffers have symmetric drive capabilities. A superbuffer consists of a push-pull or totem pole output inverter driven by a conventional inverter. In an inverting superbuffer, the gates of both pull-down transistors in the driving and the totem pole inverters are driven by the input signal whereas the gate of the pull-up transistor in the output totem pole inverter is driven by the complement of the input signal.

An inverting and a noninverting NMOS superbuffer is shown in Fig. 6.18. By designing for an inverter ratio (K_R) of 4, and driving the totem pole pull-up with twice the gate voltage of a standard depletion mode pull-up, the NMOS superbuffer can be rendered essentially ratioless. In standard NMOS inverters, the pull-up transistor has the slower switching speed. Consider the inverting superbuffer of Fig. 6.18(a). When the input voltage goes low, the output voltage of the standard inverter and the gate voltage of Q_4 goes high rapidly since the only load it sees is the small gate capacitance of Q_4. Thus, the totem pole output switches rapidly. Similar considerations shows that the noninverting super buffer also results in fast switching speeds.

The improvement in drive current capability of the NMOS superbuffer, relative to the standard (depletion load) NMOS inverter, can be estimated by comparing the average, output pull-up currents (Fabricius, 1990). The depletion load in the standard NMOS inverter is in saturation for $V_O < 2$ V and in linear region for $V_O > 2$ V. For the pull-up device, $V_{DS} = 5\,\text{V} - V_O$. Thus, the pull-up transistor is in saturation when it has $3\,\text{V} < V_{DS} < 5$ V and is in the linear region when $0\,V < V_{DS} < 3\,\text{V}$. The average current will be estimated by evaluating $I_D(\text{sat})$ at $V_{DS} = 5$ V and $I_{D(\text{lin})}$ at $V_{DS} = 2.5$ V. Let $V_{TD} = -3$ V for the depletion mode transistor. Then for the standard NMOS inverter

$$I_{D(\text{sat})} = K_{\text{pu}}(V_{GS} - V_{\text{TD}})^2 = K_{\text{pu}}(0+3)^2 = 9K_{\text{pu}} \tag{6.21}$$

$$I_{D(\text{lin})} = K_{\text{pu}}\left(2(V_{GS} - V_{\text{TD}})V_{DS} - V_{DS}^2\right) = K_{\text{pu}}(2(0+3)2.5 - 2.5^2) = 8.76K_{\text{pu}} \tag{6.22}$$

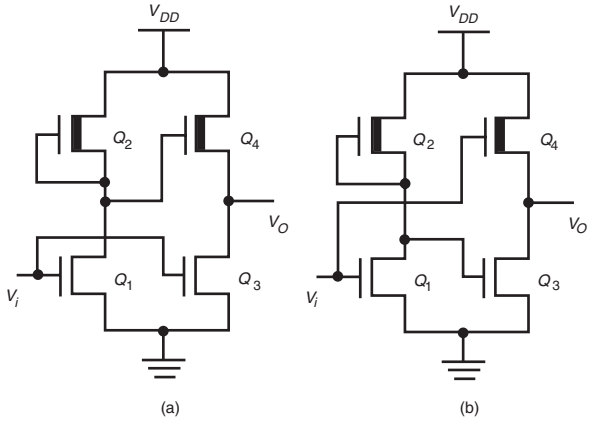


FIGURE 6.18 NMOS superbuffers: (a) inverting superbuffer, (b) noninverting superbuffer.

© 2006 by Taylor & Francis Group, LLC

Thus the average pull-up current for the standard NMOS inverter is approximately $8.88K_{\text{pu}}$. For the totem pole output of the NMOS superbuffer, the average pull-up current is also estimated from drain currents at $V_{DS} = 5$ V and 2.5 V. Note that in this case, the pull-up transistor has $V_G = V_{DD} = 5$ V when it is on. Thus, $V_{GS} = V_{DS} = 5$ V so that it always operates in the linear region. The currents are

$$I_D(5\text{ V}) = K_{\text{pu}}[2(5+3)5 - 5^2] = 55K_{\text{pu}} \tag{6.23}$$

$$I_D(2.5\text{ V}) = K_{\text{pu}}(2(2.5+3)2.5 - 2.5^2) = 10.62K_{\text{pu}} \tag{6.24}$$

The average pull-up current for the totem pole output is $38.12K_{\text{pu}}$. The average totem pole pull-up current is approximately 4.3 times the average NMOS pull-up current. Consequently, the superbuffer is roughly ratioless if designed for an inverter ratio of 4.

6.11 Output Buffers

Internal gates on a VLSI chip have load capacitances of about 50 fF or less and typical propagation delays of less than 1 ns. However, the chip output pins have to drive large capacitive loads of about 50 pF or more (Hodges and Jackson, 1988). For MOSFETs, the propagation delay is directly proportional to load capacitance. Thus, using a typical gate on the chip to drive an output pin would result in too long a propagation delay. Output buffers utilize a cascade of inverters of progressively larger drive capability to reduce the propagation delay.

An N-stage output buffer is illustrated in Fig. 6.19. Higher drive capability results from employing transistors of increasing channel width. As the transistor width increases from stage to stage by a factor of f, so does the current drive capability and the input capacitance. If C_G is the input or gate capacitance of the first inverter in the buffer, then the second inverter has an input capacitance of fC_G and the Nth inverter has an input capacitance of $f^{N-1}C_G$ and a load capacitance of $f^N C_G$, which is equal to C_L, the load capacitance at the output pin. The inverter on the left in the figure is a typical inverter on the chip with an input or gate capacitance of C_G and a propagation delay of τ. The first inverter in the buffer has an input capacitance of fC_G, but it has a current driving capability f times larger than the on chip inverter. Thus, it has a propagation delay of $f\tau$. The second inverter in the buffer has an input capacitance of f^2C_G and an accumulated delay of $2f\tau$ at its output. The Nth inverter has an input capacitance of $f^N C_G$, which is equal to the load capacitance at the output pin, and an accumulated propagation delay of $Nf\tau$, which is the overall delay of the buffer.

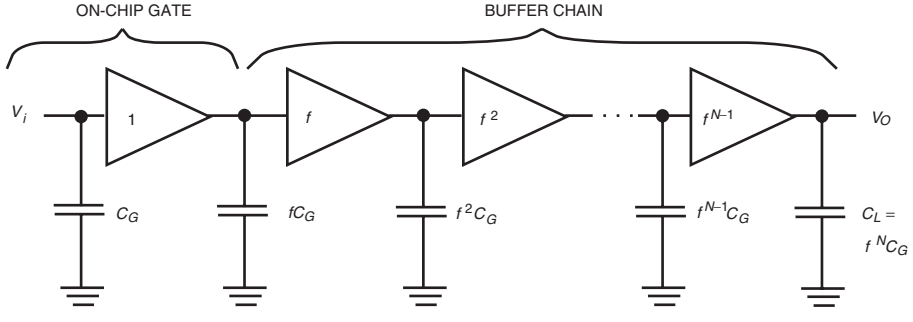


FIGURE 6.19 An N-stage output buffer chain.

© 2006 by Taylor & Francis Group, LLC

Let

$$\text{load capacitance} = YC_G = f^N C_G$$
$$\text{total delay} = t_B = Nf\tau$$

Then

$$N = \frac{1nY}{1nf} \tag{6.25}$$

and

$$t_B = \frac{1nY}{1nf} f\tau \tag{6.26}$$

By equating to zero the first derivative of t_B with respect to f, it is found that t_B is minimum at $f = e = 2.72$, the base of the natural logarithms. This is not a sharp minimum (Moshen and Mead, 1979), and values of f between 2 and 5 do not greatly increase the time delay.

Consider an example in which $C_G = 50$ fF and $\tau = 0.5$ ns for a typical gate driving an identical gate on the chip. Suppose this typical gate is to drive an output pin with load capacitance $C_L = 55$ pF, instead of an identical gate. If an output buffer is used

$$Y = \frac{C_L}{C_G} = \frac{55 \text{ pF}}{50 \text{ fF}} = 1100$$
$$N = 1nY = 7$$
$$t_B = 7e\tau = 9.5 \text{ ns}$$

If the typical chip gate is directly connected to the output pin, the propagation delay is $Y\tau = 550$ ns, which is extremely large compared with the 9.5 ns delay obtained when the buffer is used. This example illustrates the effectiveness of the buffer.

Defining Terms

Cell library: A collection of simple logic elements that have been designed in accordance with a specific set of design rules and fabrication processes. Interconnections of such logic elements are often used in semicustom design of more complex IC chips.

Custom design: A design method that aims at providing a unique implementation of the function needed for a specific application in a way that minimizes chip area and possibly other performance features.

Design rules: A prescription for preparing the photomasks used in IC fabrication so that optimum yield is obtained in as small a geometry as possible without compromising circuit reliability. They specify minimum device dimensions, minimum separation between features, and maximum misalignment of features on an IC chip.

Layout: An important step in IC chip design that specifies the position and dimension of features and components on the different layers of masks.

Masks: A set of photographic plates used to define regions for diffusion, metalization, etc., on layers of the IC wafer. Each mask consists of a unique pattern: the image of the corresponding layer.

Standard cell: A predefined logic circuit in a cell library designed in accordance with a specific set of design rules and fabrication processes. Standard cells are typically employed in semicustom design of more complex circuits.

Semicustom design: A design method in which largely predesigned subcircuits or cells are interconnected to form the desired more complex circuit or part of it.

References

Alexander, B. 1985. MOS and CMOS arrays. In *Gate Arrays: Design Techniques and Application*. ed. J.W. Read. McGraw-Hill, New York.

Fabricius, E.D. 1990. *Introduction to VLSI Design*. McGraw-Hill, New York.

© 2006 by Taylor & Francis Group, LLC

Hodges, A.D. and Jackson, H.G. 1988. *Analysis and Design of Digital Integrated Circuits*. McGraw-Hill, New York.

Horowitz, M. 1983. Timing models for MOS pass networks. *Proceedings of the IEEE Symposium on Circuits and Systems*, pp. 198–201.

Keyes, R.W. 1975. Physical limits in digital electronics. *Proc. of IEEE* 63:740–767.

Lighthart, M.M., Aarts, E.H.L., and Beenker, F.P.M. 1986. Design for testability of PLAs using statistical cooling. *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pp. 339–345, June 29–July 2.

Mavor, J., Jack, M.A., and Denyer, P.B. 1983. *Introduction to MOS LSI Design*. Addison-Wesley, Reading, MA.

Mead, C.A. and Conway, L.A. 1980. *Introduction to VLSI Systems*. Addison-Wesley, Reading, MA.

Moshen, A.M. and Mead, C.A. 1979. Delay time optimization for driving and sensing signals on high capacitance paths of VLSI systems. *IEEE J. of Solid State Circ.* SC-14(2):462–470.

Pucknell, D.A. and Eshroghian, K. 1985. *Basic VLSI Design, Principles and Applications*. Prentice-Hall, Englewood Cliffs, NJ.

Reinhard, D.K. 1987. *Introduction to Integrated Circuit Engineering*. Houghton-Mifflin, Boston, MA.

Taub, H. and Schilling, D. 1977. *Digital Integrated Circuits*. McGraw-Hill, New York.

USC Information Sciences Inst. 1984. MOSIS scalable NMOS process, version 1.0. Univ. of Southern California, Nov., Los Angeles, CA.

USC Information Sciences Inst. 1988. MOSIS scalable and generic CMOS design rules, revision 6. Univ. of Southern California, Feb., Los Angeles, CA.

Weinberger, A. 1967. Large scale integration of MOS complex logic: a layout method. *IEEE J. of Solid-State Circ.* SC-2(4):182–190.

Further Information

IEEE Journal of Solid State Circuits.

IEEE Proceedings of the Custom Integrated Circuits Conference.

IEEE Transactions on Electron Devices.

Proceedings of the European Solid State Circuits Conference (ESSCIRC).

Proceedings of the IEEE Design Automation Conference.

© 2006 by Taylor & Francis Group, LLC

7

Digital Logic Families

7.1 Introduction 7-1
7.2 Transistor-Transistor Logic 7-2
7.3 CMOS Logic 7-6
7.4 Emitter-Coupled Logic 7-10
Gallium Arsenide
7.5 Programmable Logic 7-13
Programmable Array Logic • Programmable Logic Arrays

Robert J. Feugate, Jr.

7.1 Introduction

Digital devices are constrained to two stable operating regions (usually voltage ranges) separated by a transition region through which the operating point may pass but not remain (see Fig. 7.1). Prolonged operation in the transition region does not cause harm to the devices, it simply means that the resulting outputs are unspecified. For the inverter shown in the figure, the output voltage is guaranteed to be greater than V_{OH} as long as the input voltage is below the specified V_{IH}. Note that the circuit is designed so that the input high voltage is lower than the output high voltage (vice versa for logic low voltages). This difference, called **noise margin**, permits interfering signals to corrupt the logic voltage within limits without producing erroneous operation. Logical conditions and binary numbers can be represented physically by associating one of the stable voltage ranges with one logic state or binary value and identifying the other stable voltage with the opposite state or value. By extension, then, it is possible to design electronic circuits that physically perform logical or arithmetic operations. As detailed examination of logic design is beyond the scope of this chapter, the reader is referred to any of the large number of digital logic textbooks.[1]

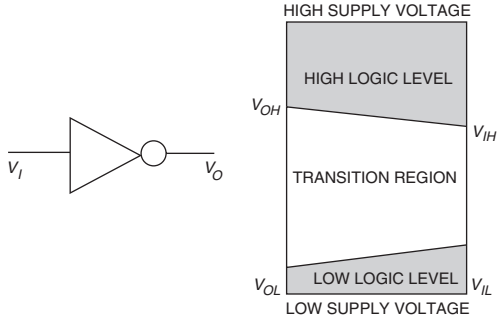


FIGURE 7.1 Logic voltage levels.

Digital logic components were the earliest commercially produced integrated circuits. *Resistor-transistor logic* (RTL) and a speedier variant *resistor-capacitor-transistor logic* (RCTL) were introduced in the early 1960s by Fairchild Semiconductor *Diode-transistor logic* (DTL) was introduced a few years later by Signetics Corporation. Although these families are often discussed in electronics texts as ancestors of later logic families, they have been obsolete for many years and are of historical interest only.

[1]Two excellent recent logic design texts are *Modern Digital Design*, by Richard Sandige, McGraw-Hill, New York, 1990 and *Contemporary Logic Design*, by Randy Katz, Benjamin Cummings, Redwood City, CA, 1994.

© 2006 by Taylor & Francis Group, LLC

TABLE 7.1 Speed-Product Comparison for Popular Discrete Logic Families[a]

Logic Family	Power Dissipation/Gate: Static at 100 kHz, mW	Typical Delay, mW	Product at 100 kHz, ns	pJ
Metal-gate CMOS	0.001	0.1	75	7.5
Silicon-gate CMOS	0.0000025	.17	10	1.7
Standard TTL	10	10	10	100
Low Power Schottky TTL	2	2	8	16
Schottky TTL	19	19	4.5	85.5
Adv. Schottky TTL	8.5	8.5	1	8.5
ALS TTL	1	1	2.5	2.5
10 K ECL	24	24	0.2	4.8

[a]Load capacitance is 50 pF, load resistance 500 Ω, except ECL is driving 50-Ω transmission.

A primary performance characteristic of different logic families is their *speed-power product*, that is, the average propagation delay of a basic gate multiplied by the average power dissipated by the gate. Table 7.1 lists the speed-power product for several popular logic families. Note that propagation delays specified in manufacturer handbooks may be measured under different loading conditions, which must be taken into account in computing speed-power product.

7.2 Transistor-Transistor Logic

First introduced in the 1960s, transistor-transistor logic (TTL) was the technology of choice for discrete logic designs into the 1990s, when complementary metal oxide semi-conductor (CMOS) equivalents gained ascendancy. Because TTL has an enormous base of previous designs, is available in a rich variety of small-scale and medium-scale building blocks, is electrically rugged, offers relatively high-operating speed, and has well-known characteristics, transistor-transistor logic continues to be an important technology for some time. Texas Instruments Corporation's 54/7400 TTL series became a *de facto* standard, with its device numbers and pinouts used by other TTL (and CMOS) manufacturers.

There are actually several families of TTL devices having circuit and semiconductor process variations that produce different speed-power characteristics. Individual parts are designated by a scheme combining numbers and letters to identify the exact device and the TTL family. Parts having 54 as the first digits have their performance specified over the military temperature range from -55 to $+125^{\circ}$C, whereas 74-series parts are specified from 0 to $+70^{\circ}$C. Letters identifying the family come next, followed by a number identifying the part's function. For example, 7486 identifies a package containing four, 2-input exclusive-OR gates from the standard TTL family, whereas 74ALS86 indicates the same function in the low-power Schottky family. Additional codes for package type and, possibly, circuit revisions may also be appended. Generally speaking, devices from different TTL families can be intermixed, although attention must be paid to **fanout** (discussed subsequently) and noise performance.

The basic building block of standard TTL is the NAND gate shown in Fig. 7.2. If either input A or B is connected to a logic-low voltage, multiple-emitter transistor Q1 is saturated, holding the base of transistor Q2 low and keeping it cut off. Consequently, Q4 is starved of base current and is also off, while Q3 receives base drive current through resistor R2 and is turned on, pulling the output voltage up toward V_{cc}. Typical high output voltage for standard TTL circuits is 3.4 V. On the other hand, if both of the inputs are high, Q1 moves into the reverse-active region and current flows out of its collector into the base of Q2. Q2 turns on, raising Q4's base voltage until it is driven into saturation. At the same time, the Ohm's law voltage drop across R2 from Q2's collector current lowers the base voltage of Q3 below the value needed to keep D1 and Q3's base-emitter junction forward biased. Q3 cuts off, with the net result that the output is drawn low, toward ground. In normal operation, then, transistors Q2, Q3, and Q4 must move between

© 2006 by Taylor & Francis Group, LLC

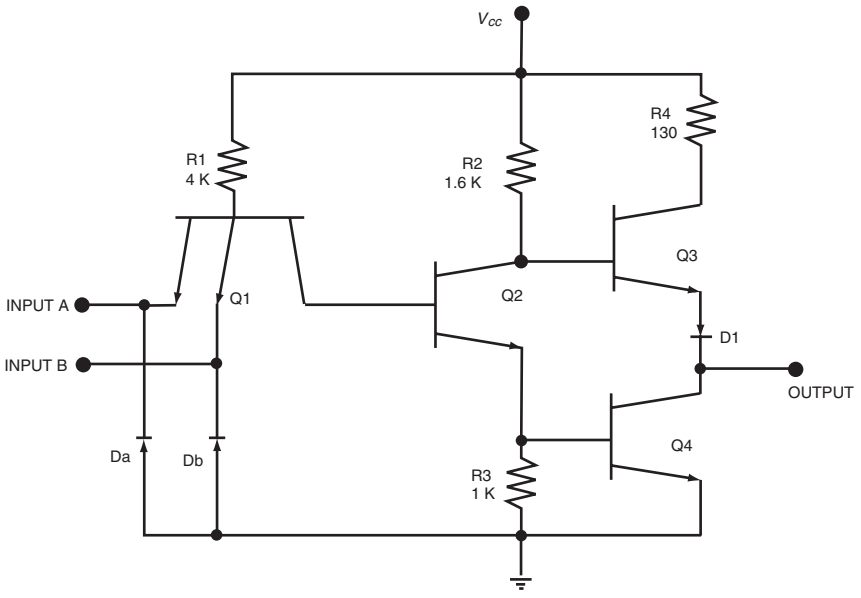


FIGURE 7.2 Two input standard TTL NAND gate.

saturation and cutoff. Saturated transistors experience an accumulation in minority carriers in their base regions, and the time required for the transistor to leave saturation depends on how quickly excess carriers can be removed from the base. Standard TTL processes introduce gold as an impurity to create *trapping sites* in the base that speed the annihilation of minority electrons. Later TTL families incorporate circuit refinements and processing enhancements designed to reduce excess carrier concentrations and speed removal of accumulated base electrons.

Because the internal switching dynamics differ and because the pull-up and pull-down transistors have different current drive capabilities, TTL parts show a longer propagation time when driving the output from low to high than from high to low (see Fig. 7.3). Although the average propagation delay $((t_{\text{phl}} + t_{\text{plh}})/2)$ is often used in speed comparisons, conservative designers use the slower propagation delay in computing circuit performance. NAND gate delays for standard TTL are typically 10 ns, with guaranteed maximum t_{phl} of 15 ns and t_{phl} of 22 ns.

When a TTL gate is producing a low output, the output voltage is the collector-to-emitter voltage of pull-down transistor Q4. If the collector current flowing in from the circuits being driven increases, the output voltage rises. The output current, therefore, must be limited to ensure that output voltage remains in the specified logic low range (less than V_{OL}). This places a maximum on the number of inputs that can be driven low by a given output stage; in other words, any TTL output has a maximum low-level fanout.

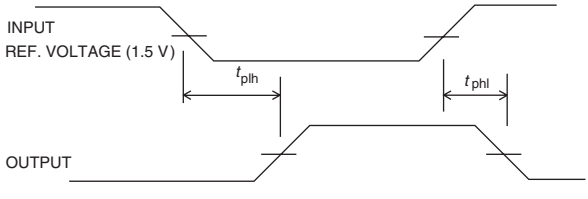


FIGURE 7.3 Propagation delay times.

© 2006 by Taylor & Francis Group, LLC

Similarly, there is a maximum fanout for high logic levels. Since circuits must operate properly at either logic level, the smaller of the two establishes the maximum overall fanout. That is

$$\text{fanout} = \text{minimum}(-I_{OH,\max}/I_{IH,\max}, -I_{OL,\max}/I_{IL,\max})$$

where

$I_{OH,\max}$ = specified maximum output current for $V_{OH,\min}$
$I_{OL,\max}$ = specified maximum output current for $V_{OL,\max}$
$I_{IH,\max}$ = specified maximum input current, high level
$I_{IL,\max}$ = specified maximum input current, low level

The minus sign in the fanout computations arises because both input and output current reference directions are defined into the device. Most parts in the standard TTL family have a fanout of 10 when driving other standard TTL circuits. Buffer circuits with increased output current capability are available for applications such as clock signals that must drive an unusually large number of inputs. TTL families other than standard TTL have different input and output current specifications; fanout computations should be performed whenever parts from different families are mixed.

One of the transistors in the conventional *totem pole* TTL output stage is always turned on, pulling the output pin toward either a logic high or low. Consequently, two or more outputs cannot be connected to the same node; if one output were attempting to drive a low logic voltage and the other a high, a low-resistance path is created from between V_{cc} and ground. Excessive currents may damage the output transistors and, even if damage does not occur, the output voltage settles in the forbidden transition region between V_{OH} and V_{OL}. Some TTL parts are available in versions with open-collector outputs, truncated output stages having only a pull down transistor (see Fig. 7.4). Two or more open-collector outputs can be connected to V_{cc} through a pull-up resistor, creating a wired-AND phantom logic gate. If any output transistor is turned on, the common point is pulled low. It only assumes a high voltage if all output transistors are turned off. The pull-up resistor must be large enough to ensure that the I_{OL} specification is not exceeded when a single gate is driving the output low, but small enough to guarantee that input leakage currents will not pull the output below V_{OH} when all driving gates are off. Since increasing the pull-up resistor value increases the resistance-capacitance (RC) time constant of the wired-AND gate and slows the effective propagation delay, the pull-up resistor is usually selected near its minimum value. The applicable design equations are

$$(V_{cc} - V_{OL,\max})/(I_{OL,\max} + nI_{IL,\max}) < R_{\text{pu}} < (V_{cc} - V_{OH,\min})/(mI_{OH,\max} + nI_{IH,\max})$$

where m is the number of open collector gates connected to R_{pu} and n is the number of driven inputs. Reference direction for all currents is into the device pin.

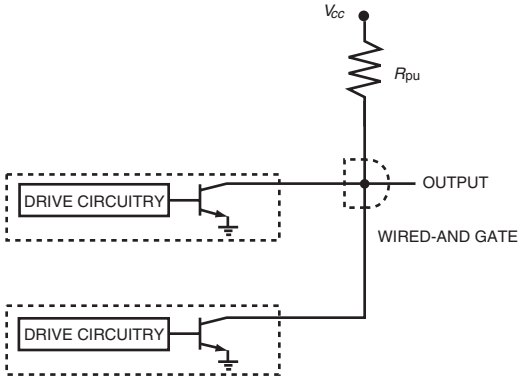


FIGURE 7.4 Wired-AND gate using open collector outputs and a pullup resistor.

© 2006 by Taylor & Francis Group, LLC

Some TTL parts have three-state outputs: totem pole outputs that have the added feature that both output transistors can be turned off under the control of an output enable signal. When the output enable is in its active state, the part's output functions conventionally, but when the enable is inactive, both transistors are off, effectively disconnecting the output pin from the internal circuitry. Several three-state outputs can be connected to a common bus, and only one output is enabled at a time to selectively place its logic level on the bus.

Over the long history of TTL, a number of variants have been produced to achieve faster speeds, lower power consumption, or both. For example, the 54/74L family of parts incorporated redesigned internal circuitry using higher resistance values to lower the supply current, reducing power consumption to a little as 1/10 that of standard TTL. Speed was reduced as well, with delay times as much as three times longer. Conversely, the 54/74H family offered higher speeds than standard TTL, at the expense of increased power consumption. Both of these TTL families are obsolete, their places taken by various forms of Schottky TTL.

As noted earlier, TTL output transistors are driven well into saturation. Rather than using gold doping to hasten decay of minority carriers, Schottky transistors limit the forward bias of the collector-base junction and, hence, excess base minority carrier accumulation by paralleling the junction with a Schottky (gold-semiconductor) diode, as shown in Fig. 7.5. As the NPN transistor starts to saturate, the Schottky diode becomes forward biased. This clamps the collector-base junction to about 0.3 V, keeping the transistor out of hard saturation. Schottky TTL (S) is about three times faster than standard TTL, while consuming about 1.5 times as much power.

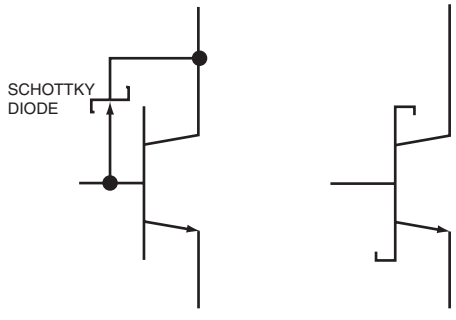


FIGURE 7.5 Schottky transistor construction and symbol.

Lower power Schottky TTL (LS) combined an improved input circuit design with Schottky transistors to reduce power consumption to a fifth of standard TTL, while maintaining equal or faster speed. The redesigned input circuitry of LS logic changes the input logic thresholds, resulting in slightly reduced noise margin.

Still another TTL family is *Fairchild Advanced Schottky* TTL (FAST). The F parts combine circuit improvements with processing techniques that reduce junction capacitances and increase transistor speeds. The result is a fivefold increase in speed compared to standard TTL, with lowered power consumption.

As their names imply, advanced Schottky (AS) and advanced low-power Schottky (ALS) use improved fabrication processes to improve transistor switching speeds. Parts from these families are two to three times faster than their S and LS equivalents, while consuming half the power. Although the standard TTL, Schottky (S), and low-power Schottky (LS) families remain in volume production, the newer advanced Schottky families (AS, ALS, F) offer better speed-power products and are favored for new designs.

In making speed comparisons between different logic families, one should keep in mind that manufacturer's published specifications use different load circuits for different TTL families. For example, propagation delays of standard TTL are measured using a load circuit of a 15-pF capacitor paralleled by a 400-Ω resistor, whereas FAST uses 50 pF and 500 Ω. Since effective propagation delay increases approximately linearly with the capacitance of the load circuit, direct comparisons of raw manufacturer data is meaningless. Manufacturer's databooks include correction curves showing how propagation delay changes with changes in load capacitance.

As outputs change from one logic state to another, sharp transients occur on the power supply current. Decoupling capacitors are connected between the V_{cc} and ground leads of TTL logic packages to provide filtering, minimizing transient currents along the power supply lines and reducing noise generation. Guidelines for distribution of decoupling capacitors are found in manufacturer handbooks and texts on digital system design, for example, Barnes (1987).

© 2006 by Taylor & Francis Group, LLC

Circuit boards using advanced TTL with very short **risetimes** (FAST and AS) require physical designs that reduce waveform distortion due to transmission line effects. A brief discussion of such considerations is given in the following section on emitter-coupled logic.

7.3 CMOS Logic

In 1962, Frank Wanlass of Fairchild Semiconductor noted that enhancement nMOS and pMOS transistors could be stacked totem pole fashion to form an extraordinarily simple inverter stage (see Fig. 7.6). If the input voltage is near V_{dd}, the nMOS pull-down transistor channel will be enhanced, whereas the pMOS transistor remains turned off. The voltage divider effect of a low nMOS channel resistance and very high pMOS channel resistance produces a low output voltage. On the other hand, if the input voltage is near ground, the nMOS transistor will be turned off, while the pMOS channel is enhanced to pull the output voltage high. Logic gates can be easily created by adding parallel pull-down transistors and series pull-up transistors. When a CMOS gate is in either of its static states, the only current flowing is the very small leakage current through the off transistor plus any current needed by the driven inputs. Since the inputs to CMOS gates are essentially capacitances, input currents can be very small (on the order of 100 pA). Thus, the static power supply current and power dissipation of CMOS gates is far smaller than that of TTL gates. Since the early versions of discrete CMOS logic were much slower than TTL, CMOS was usually relegated to applications where power conservation was more important than performance.

In TTL logic, the high and low output logic levels and the input switching threshold (point at which a particular gate input interprets the voltage as a one rather than a zero) are established largely by forward bias voltages of PN junctions. Since these voltages do not scale with the power supply voltage, TTL operates properly over a rather restricted range (5.0 V $\pm$ 10%). In an all CMOS system, however, output voltage is determined by the voltage divider effect. Since the off-resistance of the inactive transistor's channel is far larger than the resistance of the active channel, CMOS logic high and low levels both are close to the power supply values (V_{dd} and ground) and scale with power supply voltage. The output transistors are designed to have matching channel resistances when turned on. Thus, as long as the supply voltage remains well above the transistor threshold voltage (about 0.7 V for modern CMOS), the input switching voltage also scales with supply voltage and is about one-half of V_{dd}. Unlike TTL, CMOS parts can operate over a very wide range of supply voltages (3–15 V) although lower voltages reduce noise immunity and speed. In the mid-1990s, integrated circuit manufacturers introduced new generations of CMOS parts optimized for operation in the 3-V range, intended for use in portable applications powered by two dry cell batteries.

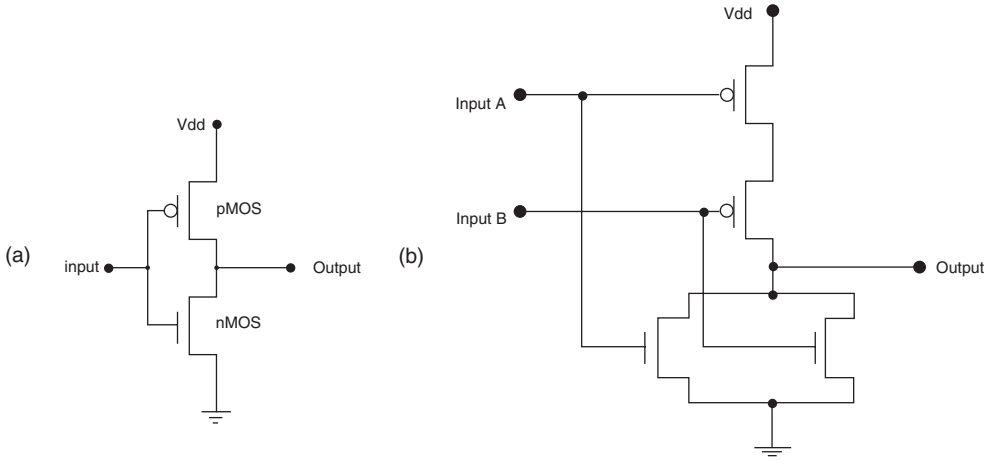


FIGURE 7.6 Elementary CMOS logic circuit: (a) complementary MOS inverter stage, (b) two-input CMOS NOR gate.

© 2006 by Taylor & Francis Group, LLC

Requiring only four transistors, the basic CMOS gate requires much less area than an LS TTL gate with its 11 diodes and bipolar transistors and 6 diffused resistors. This fact, coupled with its much higher static power consumption, disqualifies TTL as a viable technology for very large-scale integrated circuits. In the mid-1980s, CMOS quickly obsoleted the earlier nMOS integrated circuits and has become almost the exclusive technology of choice for complex digital integrated circuits.

Older discrete CMOS integrated circuits (RCA's 4000-series is the standard, with several manufacturers producing compatible devices) use aluminum as the gate material. The metal gate is deposited after creation of the drain and source regions. To guarantee that the gate overlies the entire channel even at maximum interlayer alignment tolerances, there is a substantial overlap of the gate and drain and source. Comparatively large parasitic capacitances result, slowing transistor (hence, gate) speed. In the mid-1980s several manufacturers introduced advanced discrete CMOS logic based on the self-aligning polysilicon gate processes developed for large-scale integrated circuits (ICs). Discrete silicon gate parts are often functional emulations of TTL devices, and have similar numbering schemes (e.g., 74HCxxx). Since silicon gate CMOS is much faster than metal gate CMOS (8 ns typical gate propagation delay vs 125 ns.) while offering similar static power consumption (1 mW per gate vs. 0.6 mW), silicon gate CMOS is the technology of choice for new discrete logic designs. There are two distinct groups of modern CMOS parts: HC high-speed CMOS and AC or C advanced CMOS.

Even though parts from the 74HC and similar families may emulate TTL functions and have matching pin assignments, they cannot be intermingled with TTL parts. As mentioned, CMOS output voltages are much closer to the power supply voltages than TTL outputs. Gates are designed with this characteristic in mind and do not produce solid output voltages when driven by typical TTL high logic levels. Specialized advanced CMOS parts are produced with modified input stages specifically for interfacing with TTL circuits (for example, the HCT family). These parts exhibit slightly higher power consumption than normal silicon-gate CMOS.

The gate electrodes of CMOS transistors are separated from the channel by a layer of silicon dioxide that may be only a few hundred angstroms thick. Such a thin dielectric layer can be damaged by a potential difference of only 40–100 V. Without precautions, the normal electrostatic potentials that build up on the human body can destroy CMOS devices. Although damage from electrostatic discharge (ESD) can be minimized through proper handling procedures (Matisof, 1986), CMOS parts include ESD protection circuitry on all inputs (see Fig. 7.7). The input diodes serve to limit gate voltages to one diode drop above V_{dd} or below ground, whereas the resistor serves both to limit input current and to slow the rise time of very fast pulses.

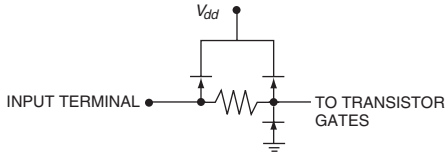


FIGURE 7.7 A typical CMOS input protection circuit.

As shown in Fig. 7.8, CMOS inverters and gates inherently have cross-connected parasitic bipolar transistors that form a silicon controlled rectifier (SCR). Suppose that external circuitry connected to the output pin pulls the pin low, below ground level. As this voltage approaches −0.5 V, the base-emitter junction of the parasitic NPN transistor will start to forward bias and the transistor will turn on. The resulting collector current is primarily determined by whatever external circuitry is pulling the output low. That current flows through R_{well} and lowers the base voltage of the parasitic PNP transistor. A large enough current will forward bias the PNP base emitter junction, causing PNP collector current to flow through R_{sub} and helping to maintain the forward bias of the NPN transistor. The SCR then enters a regenerative condition and will quickly assume a stable state in which both transistors remain on even after the initial driving stimulus is removed. In this **latchup** condition, substantial current flows from V_{dd} to ground. Normal operation of the CMOS gate is disrupted and permanent damage is likely. For latchup to occur, the product of the NPN and PNP transistor current gains must be greater than one, the output pin must be driven either below −0.5 V or above $V_{dd} + 0.5$ V by a source that supplies enough current to trigger regenerative operation, and V_{dd} must supply a sustaining current large enough to keep the SCR turned on. Although the potential for latchup cannot be avoided, CMOS manufacturers design input and output

© 2006 by Taylor & Francis Group, LLC

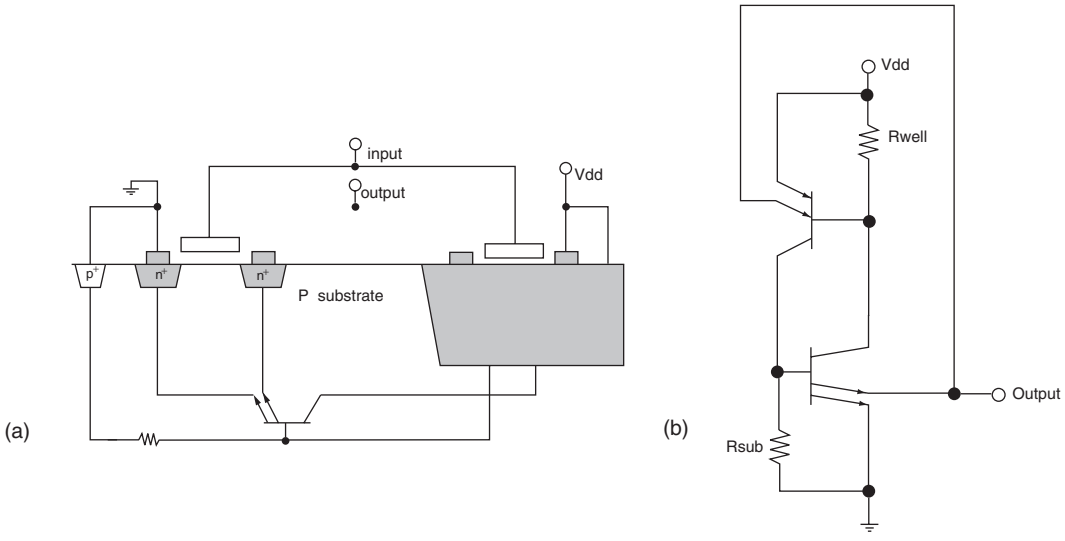


FIGURE 7.8 CMOS parasitic SCR: (a) cross-section showing parasitic bipolar transistors, (b) equivalent circuit.

circuits that are latchup resistant, using configurations that reduce the gain of the parasitic transistors and that lower the substrate and *p*-well resistances. Nevertheless, CMOS users must ensure that input and output pins are not driven outside normal operating ranges. Limiting power supply current below the holding current level is another means of preventing latchup damage.

Fanout of CMOS outputs driving TTL inputs is limited by static current drive requirements and is calculated as for transistor-transistor logic. In all-CMOS systems, the static input current is just the leakage current flowing through the input ESD protection diodes, a current on the order of 100 pA. Consequently, static fanout of CMOS driving CMOS is so large as to be meaningless. Instead, fanout is limited by speed deterioration as the number of driven gates increases (Texas Instruments, 1984). If all driven inputs are assumed identical, an all-CMOS system can be represented by the approximate equivalent circuit of Fig. 7.9. The parallel input resistance R_{in} is on the order of 60 MΩ and can be neglected in later computations. The pull-up resistor R_{pu} can be approximately calculated from manufacturer data as $(V_{dd} - V_{OH})/I_{OH} (\sim 50\,\Omega)$. The output will charge exponentially from low to high with a time constant of $nR_{\text{pu}}C_{\text{in}}$, where n is the actual fanout. Since the low output voltage is nearly zero, time required to reach a logic high is given by

$$t = nR_{\text{pu}}C_{\text{in}} \ln(1 - V_{IH,\text{min}}/V_{OH}).$$

Thus, once the maximum allowable risetime (that is, the maximum effective propagation delay) has been specified for an all-CMOS system, the dynamic fanout n can be computed. Although static power

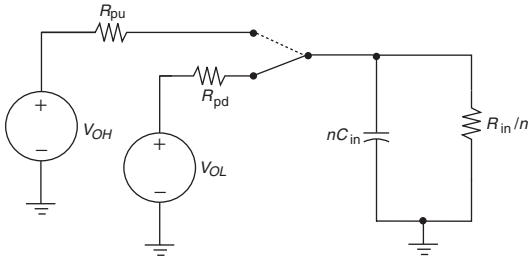


FIGURE 7.9 Equivalent circuit for a CMOS output driving *n*CMOS inputs.

© 2006 by Taylor & Francis Group, LLC

consumption of CMOS logic is quite small, outputs that are changing logic states will dissipate power each time they charge and discharge the capacitance loading the output. In addition, both the pull-up and pull-down transistors of the output will be simultaneously partially conducting for a short period on each switching cycle, creating a transient supply current beyond that needed for capacitance charging. Thus, the total power consumption of a CMOS gate is given by the relation

$$P_{\text{total}} = P_{\text{static}} + C_{\text{load}} V_{dd}^2 f$$

where

P_{total} = total power dissipated
P_{static} = static power dissipation
C_{load} = total equivalent load capacitance, $C_{\text{pd}} + C_{\text{ext}}$
C_{pd} = *power dissipation capacitance*, a manufacturer specification representing the equivalent internal capacitance (the effect of transient switching current is included)
C_{ext} = total parallel equivalent load capacitance
V_{dd} = power supply voltage
f = output operating frequency

HCT and ACT parts include an additional static power dissipation term proportional to the number of TTL inputs begin driven ($n\Delta I_{dd} V_{dd}$, where ΔI_{dd} is specified by the manufacturer). CMOS average power consumption is proportional to the frequency of operation, whereas power consumption of TTL circuits is almost independent of operating frequency until about 1 MHz, when dynamic power dissipation becomes significant (see Fig. 7.10). Direct comparison of CMOS and TTL power consumption for a digital system is difficult, since not all outputs are changing at the same frequency. Moreover, the relative advantages of CMOS become greater with increasing part complexity; CMOS and TTL power curves for a modestly complicated decoder circuit crossover at a frequency 10 times that of simple gates. Finally, TTL manufacturers do not publish C_{pd} specifications for their parts.

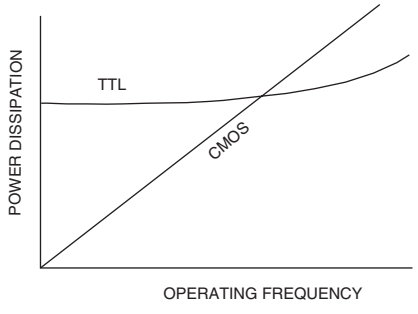


FIGURE 7.10 Power dissipation vs. operating frequency for TTL and CMOS logic devices.

Like TTL parts, CMOS logic requires use of decoupling capacitors to reduce noise due to transient current spikes on power supply lines. Although such decoupling capacitors reduce electromagnetic coupling of noise into parallel signal leads, they do not eliminate an additional noise consideration that arises in both advanced CMOS and advanced Schottky due to voltages induced on the power supply leads from transient current spikes. Consider the multiple inverter IC package of Fig. 7.11, in which five outputs are switching from low to high simultaneously, while the remaining output stays low. In accord with Faraday's law, the switching current spike will induce transient voltage across L, which represents parasitic inductances from internal package leads and external ground supply traces. That same voltage also lifts the unchanging output above the ground reference. A sufficiently large voltage spike may make signal A exceed the input switching threshold and cause erroneous operation of following logic elements. Careful physical design of the power supply and ground leads can reduce external inductance and limit power supply sag and ground bounce. Other useful techniques include use of synchronous clocking and segregation of high drive signals, such as clock lines, into separate buffer packages, separated from reset and other asynchronous buffers.

As has been suggested several times, effective propagation delay of a logic circuit is directly proportional to the capacitive loading of its output. Systems built from small- and medium-scale logic have many integrated circuit outputs with substantial load capacitance from interconnecting printed circuit traces. This is the primary impetus for using larger scale circuits: on-chip interconnections present far smaller capacitance and result in faster operation. Thus, a system built using a highly integrated circuit

© 2006 by Taylor & Francis Group, LLC

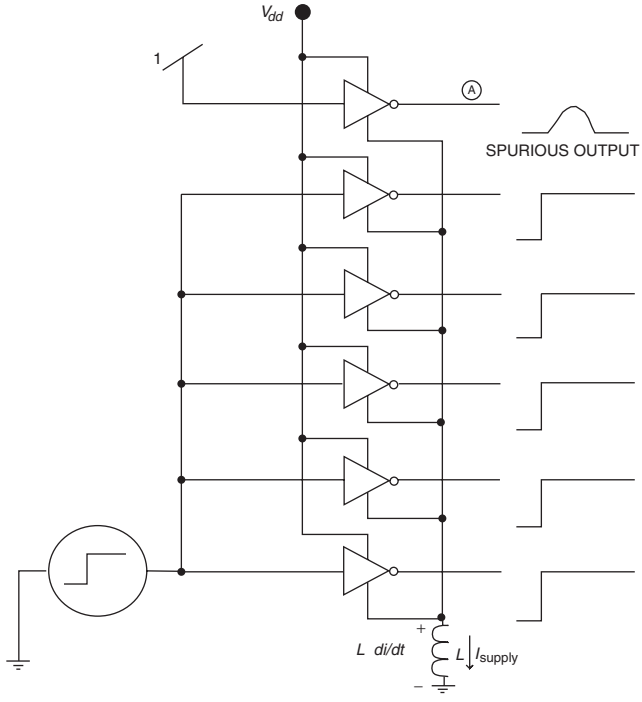


FIGURE 7.11 Output bounce due to ground lead inductance.

fabricated in a slower technology may be faster overall that the same system realized using faster small-scale logic.

Bipolar-CMOS (BiCMOS) logic is a hybrid technology for very large-scale integrated circuits that incorporates both bipolar and MOS transistors on the same chip. It is intended to offer the most of the circuit density and static power savings of CMOS while providing the electrical ruggedness and superior current drive capability of bipolar logic. BiCMOS also holds the opportunity for fabricated mixed systems using bipolar transistors for linear portions and CMOS for logic. It is a difficult task to produce high-quality bipolar transistors on the same chip as MOS devices, meaning that biCMOS circuits are more expensive and may have suboptimal performance.

7.4 Emitter-Coupled Logic

Emitter coupled logic (ECL) is the fastest variety of discrete logic. It is also one of the oldest, dating back to Motorola's MECL I circuits in 1962. Improved MECL III circuits were introduced in the late 1960s. Today, the 10 and 100 K ECL families remain in use where very high speed is required. Unlike either TTL or CMOS, emitter-coupled logic uses a differential amplifier configuration as its basic topology (see Fig. 7.12). If all inputs to the gate are at a voltage lower than the reference voltage, the corresponding input transistors are cutoff and the entire bias current flows through Q1. The output is pulled up to V_{cc} through resistor R_a. If any input voltage rises above the reference, the corresponding transistor turns on, switching the current from Q1 to the input transistor and lowering the output voltage to $V_{cc} - I_{\text{bias}} R_a$. If the collector resistors and bias current are properly chosen, the transistors never enter saturation, and switching time is quite fast. Typical propagation delay for MECL 10 K gates is only 2.0 ns. To reduce the output impedance and improve current drive capability, actual ECL parts include emitter follower outputs (Fig. 7.12(b)). They also include integral voltage references with temperature coefficients matching those of the differential amplifier.

© 2006 by Taylor & Francis Group, LLC

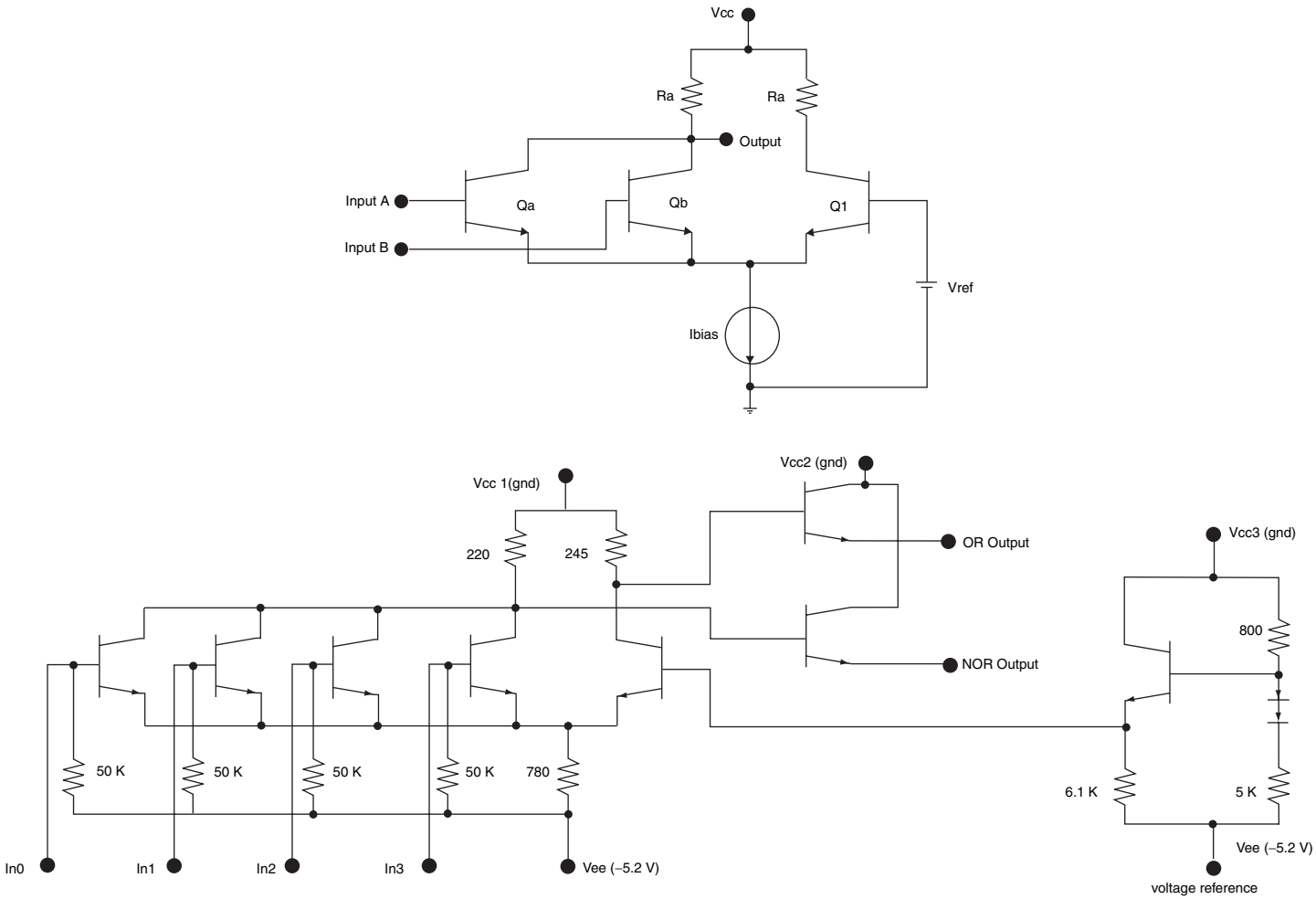


FIGURE 7.12 Emitter coupled logic (ECL) gates: (a) simplified schematic of ECL NOR gate, (b) commercial ECL gate.

© 2006 by Taylor & Francis Group, LLC

Note that ECL circuits are normally operated with the collectors connected to ground and the emitter current source to a negative supply voltage. From the standpoint of supply voltage fluctuations, the circuit operates as a common base amplifier, attenuating supply variations at the output. The power supply current in both TTL and CMOS circuits exhibits sharp spikes during switching; in ECL gates, the bias current remains constant and simply shifts from one transistor to another. Lower logic voltage swings also reduce noise generation. Although the noise margin for ECL logic is less than TTL and CMOS, ECL circuits generate less noise.

Emitter-coupled logic offers both short propagation delay and fast risetimes. When interconnecting paths introduce a time delay longer than about one-half the risetime, the interconnection's transmission line behavior must be taken into account. Reflections from unterminated lines can distort the waveform with ringing. The need to wait until reflections have damped to an acceptable value increases the effective propagation delay. The maximum open line length depends on the fanout of the driving output, the characteristic impedance of the transmission line, and the velocity of propagation of the line. Motorola databooks indicate that MECL III parts, with an actual fanout of 8 connected with fine-pitch printed circuit traces (100-Ω microstrip), can tolerate an unterminated line length of only 0.1 in. In high-speed ECL designs, physical interconnections must be made using properly terminated transmission lines (see Fig. 7.13). For an extended treatment of transmission lines in digital systems, see manufacturer's publications (Blood, 1988) or Rosenstark (1994). The necessity for using transmission lines with virtually all ECL III designs resulted the introduction in 1971 of the MECL 10 K family of parts, with 2-ns propagation delays and 3.5-ns risetimes. The slowed risetimes meant that much longer unterminated lines could be used (Motorola cites 2.6 in. for the configuration mentioned earlier). Thus, many designs could be implemented in 10 K ECL without transmission line interconnections. This relative ease of use and a wider variety of parts has made 10 K ECL and its variants the dominant ECL family. It should be noted that, although transmission line interconnections were traditionally associated with ECL, advanced CMOS and advanced Schottky TTL have rapid risetimes and are being used in very high-speed systems. Transmission line effects must be considered in the physical design of any high-performance system, regardless of the logic family being used.

Although ECL is often regarded as a power-hungry technology compared to TTL and CMOS, this is not necessarily true at very high-operating frequencies. At high frequencies, the power consumption for both

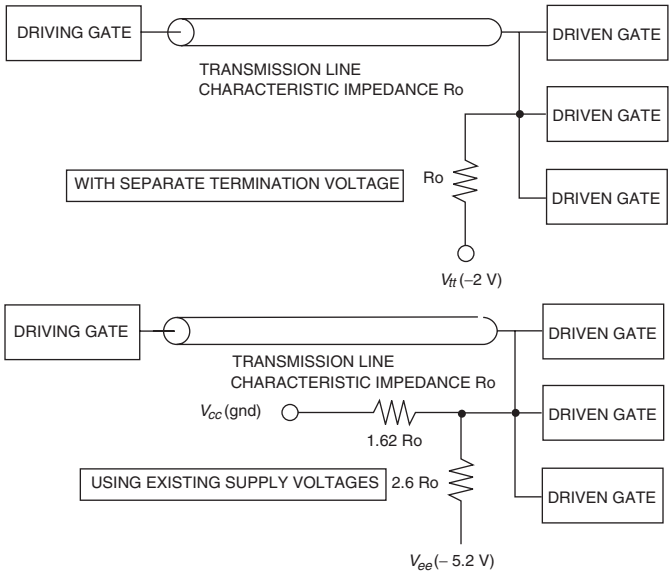


FIGURE 7.13 Parallel-terminated ECL interconnections.

© 2006 by Taylor & Francis Group, LLC

of the latter families increases approximately linearly with frequency, whereas ECL consumption remains constant. Above about 250 MHz, CMOS power consumption exceeds that of ECL.

Gallium Arsenide

It was known even before the advent of the transistor that gallium arsenide constituted a semiconducting compound having significantly higher electron mobility than silicon. Holes, however, are substantially slower in GaAs than in silicon. As a consequence, unipolar GaAs transistors using electrons for charge transport are faster than their silicon equivalents and can be used to create logic circuits with very short propagation delays. Since gallium arsenide does not have a native oxide, it is difficult to fabricate high-quality MOS field effect transistors (MOSFETs). The preferred GaAs logic transistor is the *metal-semiconductor field-effect transistor* (MESFET) which is essentially a junction field effect transistor in which the gate is formed by a metal-semiconductor (Schottky) junction instead of a p–n diode. Figure 7.14 shows a NOR gate formed using two enhancement MESFETs as pull-down transistors with a depletion MESFET active load for the pull-up. At room temperatures, the underlying GaAs substrate has a very high resistivity (is semi-insulating), providing sufficient transistor-to-transistor isolation without the need for reverse-biased isolation junctions.

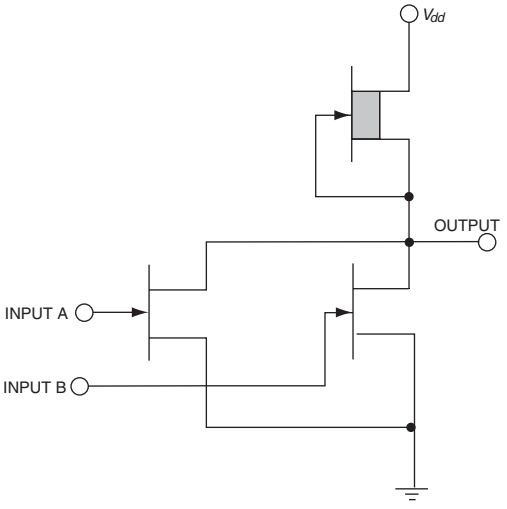


FIGURE 7.14 A GaAs enhancement/depletion-mode MESFET NOR gate.

Gallium arsenide logic exhibits both short delay and fast risetimes. As noted in the ECL discussion, fast logic signals exhibit ringing if transmission line interconnections are not used. The waveform distortion due to ringing extends the effective propagation delay and can negate the speed advantage of the basic logic circuit. Furthermore, complex fabrication processes and limited volumes make gallium arsenide logic from 10 to 20 times more expensive than silicon. The dual requirements to control costs and to preserve GaAs speed advantage by limiting the number of off-chip interfaces make the technology better suited to large-scale rather small- and medium-scale logic. GaAs digital circuits are available from several vendors in the form of custom integrated circuits or gate arrays. Gate arrays are semifinished large-scale integrated circuits that have large numbers of unconnected gates. The customer specifies the final metallization patterns that interconnect the gates to create the desired logic circuits. As of this writing, vendors were offering application-specific GaAs integrated circuits with more than 20,000 gates and worst-case internal gate delays on the order of 0.07 ns.

7.5 Programmable Logic

Although traditional discrete logic remains an important means of implementing digital systems, various forms of programmable logic have become increasingly popular. Broadly speaking, programmable logic denotes large- and very large-scale integrated circuits containing arrays of generalized logic blocks with user-configurable interconnections. In programming the device, the user modifies it to define the functions of each logic block and the interconnections between logic blocks. Depending on the complexity of the programmable logic being used, a single programmable package may replace dozens of conventional small- and medium-scale integrated circuits. Because most interconnections are on the programmable logic chip itself, programmable logic designs offer both speed and reliability gains over conventional logic. Effectively all programmable logic design is done using electronic design automation (EDA) tools

© 2006 by Taylor & Francis Group, LLC

that automatically generate device programming information from a description of the function to be performed. The functional description can be supplied in terms of a logic diagram, logic equations, or a hardware description language. EDA tools also perform such tasks as logic synthesis, simplification, device selection, functional simulation, and timing/speed analysis. The combination of electronic design automation and device programmability makes it possible to design and physically implement digital subsystems very quickly compared to conventional logic.

Although the definitions are somewhat imprecise, programmable logic is divided into two broad categories: programmable logic devices (PLDs) and field programmable gate arrays (FPGAs). The PLD category itself is often subdivided into complex PLDs (CPLDs) with numerous internal logic blocks and simpler PLDs. Although the general architecture of small PLDs is fairly standardized, that of FPGAs and CPLDs continues to evolve and differs considerably from one manufacturer to another. Programmable logic devices cover a very wide range, indeed, from simple PALs containing a few gates to field programmable gate arrays providing thousands of usable gate equivalents.

Programmable Array Logic

Programmable array logic (PAL) was first introduced by Monolithic Memories, Inc. In their simplest form, PALs consist of a large array of logic AND gates fed by both the true (uninverted) and complemented (inverted) senses of the input signals. Any AND gate may be fed by either sense of any input. The outputs of the AND gates are hardwired to OR gates that drive the PAL outputs (see Fig. 7.15). By programming the AND inputs, the PAL user can implement any logic function provided that the number of product terms does not exceed the **fanin** to the OR gate. Modern PALs incorporate several logic macrocells that include a programmable inverter following the AND/OR combinatorial logic, along with a programmable flip-flop that can be used when registered outputs are needed. Additional AND gates are used to clock and reset the flip-flop. Macrocells usually also have a programmable three-state buffer driving the block output and programmable switches that can be used to feed macrocell outputs back into the AND–OR array to develop combinatorial logic with many product terms or to implement state machines.

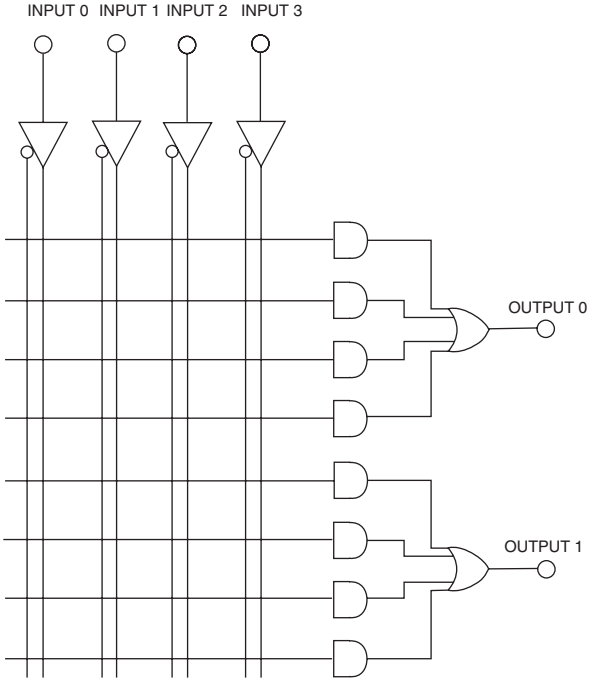


FIGURE 7.15 A simplified small PAL device.

© 2006 by Taylor & Francis Group, LLC

PLDs are fabricated using either bipolar or CMOS technologies. Although CMOS parts have become quite fast, bipolar PALs offer higher speed (<5 ns input to output combinatorial delay) at the expense of higher power consumption. Different manufacturers accomplish interconnect and logic options programming in different ways. In some instances, the circuit is physically modified by blowing fuses or antifuses (an antifuse is an open circuit until fused, then it becomes a low-resistance circuit). CMOS programmable devices often use interconnections based on erasable programmable read only memory (EPROM) cells. EPROM-based parts can be programmed, then erased and reprogrammed. Although reprogrammability can be useful during the development phase, it is less important in production, and EPROM-based PLDs are also sold in cheaper, one-time programmable versions. Although not a major market factor, electrically reprogrammable PLDs are also available. When used in digital systems, PLDs require fanout computation, just as discrete TTL or CMOS logic. In addition, fast PLDs require the same consideration of transmission line effects.

Although it is possible to hand-develop programming tables for very simple PALs, virtually all design is now done using EDA tools supplied by PLD vendors and third parties. Although computer methods automate the design process to a considerable extent, they usually permit some user control over resource assignments. The user needs to have a thorough knowledge of a specific device's architecture and programming options to optimize the resulting design.

Programmable Logic Arrays

Field programmable logic arrays (FPLAs) resemble PALs with the additional feature that the connections from the AND intermediate outputs to the OR inputs are also programmable, rather than hardwired (Fig. 7.15). Like simpler PALs, FPLAs are produced in registered versions that have flip-flops between the OR gate and the output pin. Even though they are more complex devices, FPLAs were actually introduced before programmable array logic. Programmable AND/OR interconnections make them more flexible than PALs, since product terms (AND gate outputs) can drive more than one OR input and any number of product terms can be used to develop a particular output function. The additional layer of programmable interconnects consumes chip area, however, increasing cost. The extra level of interconnections also makes programming more complicated. The most important drawback to FPLAs, however, is lower speed. The programmable interconnections add significant series resistance, slowing signal propagation between the AND and OR gates and making FPLAs sluggish compared to PALs. Since programmable array logic is faster, simpler to program, and flexible enough for most uses, PALs are much more popular than field programmable logic arrays. Most FPLAs marketed today have registered outputs (that is, a flip-flop is placed between the OR gate and the macrocell output) that are fed back into to the AND/OR array. Termed *field programmable logic sequencers*, these parts are intended for implementing small state machines.

Field Programmable Gate Arrays and Complex PLDs

There is no universally accepted dividing line between complex PLDs and FPGAs. Not only do architectures differ from manufacturer to manufacturer, the method of interconnect programming also differs. The term *complex programmable logic device* (CPLD) generally denotes devices organized as large arrays of PAL macrocells, with fixed delay feedback of macrocells outputs into the AND–OR array (Fig. 7.16). CLPD time delays are more predictable than those of FPGAs, which can make them simpler to design with. A more important distinction between CPLDs and FPGAs is the ratio of combinatorial logic to sequential logic in their logic cells. The PAL-like structure of CPLDs means that the flip-flop in each logic macrocell is fed by the logical sum of several logical products. The products can include a large number of input and feedback signals. Thus, CPLDs are well suited to applications such as state machines that have logic functions including many variables. FPGAs, on the other hand, tend toward cells that include a flip-flop driven by a logic function of only three or so variables and are better suited to situations requiring many registers fed by relatively simple functions.

Field programmable gate arrays and complex programmable logic devices concentrate far more logic per package than traditional discrete logic and smaller PLDs, making computer aided design essential. The overall procedure is as follows.

© 2006 by Taylor & Francis Group, LLC

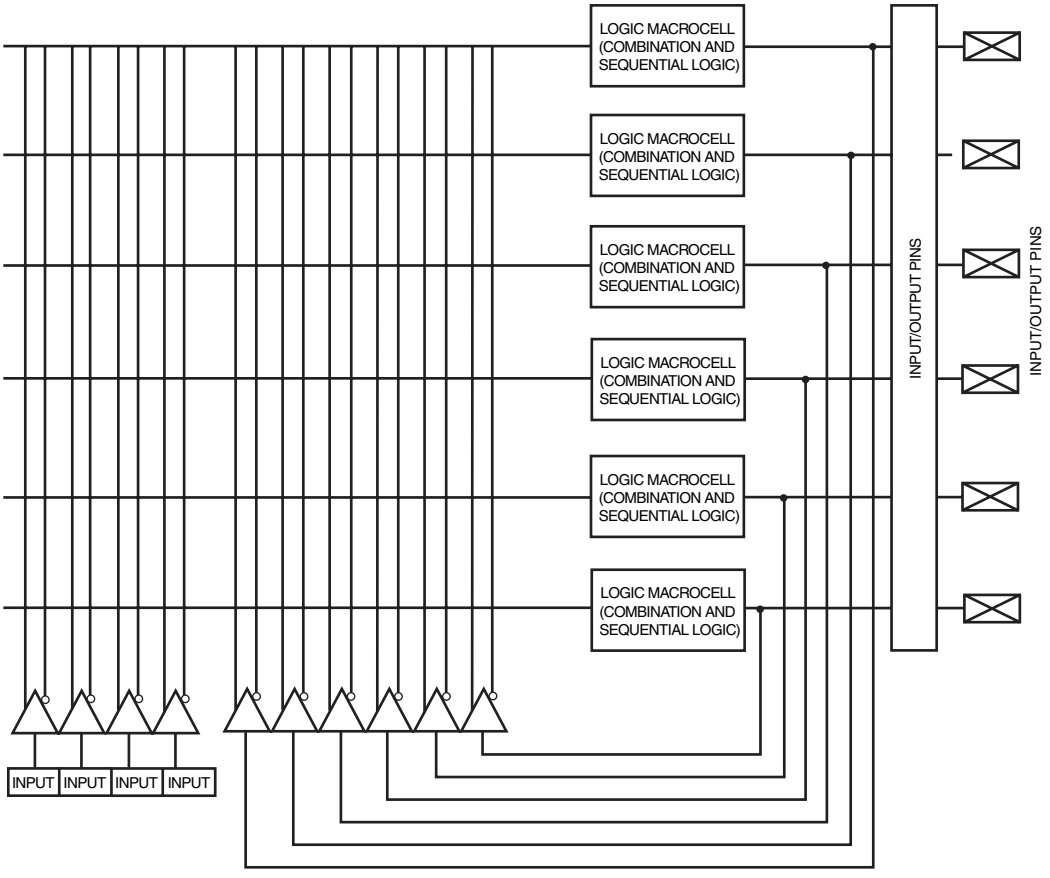


FIGURE 7.16 A simplified complex programmable logic device structure.

1. Describe system functionality to the EDA system, using schematics, boolean equations, truth tables, or high level programming languages. Subparts of the overall system may be described separately, using whatever description is most suitable to that part, and then combined. As systems become more complex, high level languages have become increasingly useful. Subsequent steps in the design process are largely automated functions of the EDA software.
2. Translate the functional description into equivalent logic cells and their interconnections.
3. Optimize the result of step 2 to produce a result well suited to the implementation architecture. The goal of the optimization and the strategies involved depend strongly on the target architecture. For instance, EDA tools for FPGAs with limited interconnections seek to reduce the number of block interconnects.
4. Fit the optimized design to the target architecture. Fitting includes selection of an appropriate part, placing of logic (mapping of functional blocks to specific physical cells), and routing of cell interconnections.
5. Produce tables containing information required for device programming and back annotate interconnection netlists. Back annotation provides information about interconnection paths and fanout that is necessary to produce accurate timing estimates during simulation.
6. Perform functional and performance verification through simulation. Simulation is critical, because of the high ratio of internal logic to input–output ports in FPGA/CPLD designs. Logic errors due to timing problems or bugs in the compilation and fitting steps can be extremely difficult to locate

© 2006 by Taylor & Francis Group, LLC

and correct based purely on performance testing of a programmed device. Simulation, on the other hand, produces invaluable information about internal behavior.

Defining Terms

Fanin: The number of independent inputs to a given logic gate.

Fanout: (1) The maximum number of similar inputs that a logic device output can drive while still meeting output logic voltage specifications. (2) The actual number of inputs connected to a particular output.

Latchup: A faulty operating condition of CMOS circuits in which its parasitic SCRs produce a low resistance path between power supply rails.

Noise margin: The difference between an output logic high (low) voltage produced by a properly functioning output and the input logic high (low) voltage required by a driven input; noise margin provides immunity to a level of signal distortion less than the margin.

Risetime: The time required for a logic signal to transition from one static level to another; usually, risetime is measured from 10 to 90% of the final value.

References

Altera. 1993. *Data Book*. Altera Corp., San Jose, CA.

Alvarez, A. 1989. *BiCMOS Technology and Applications*. Kluwer Academic, Boston, MA.

Barnes, J.R. 1987. *Electronic System Design: Interference and Noise Control Techniques*. Prentice-Hall, Englewood Cliffs, NJ.

Blood, W.R. 1988. *MECL System Design Handbook*. Motorola Semiconductor Products, Phoenix, AZ.

Brown, S.D., Francis, R., Rose, J., and Vranesic, Z. 1992. *Field-Programmable Gate Arrays*. Kluwer Academic, Boston, MA.

Buchanan, J. 1990. *CMOS/TTL Digital System Design*. McGraw-Hill, New York.

Deyhimy, I. 1985. GaAs digital ICs promise speed and lower cost. *Computer Design* 35(11):88–92.

Jenkins, J.H. 1994. *Designing with FPGAs and CPLDs*. Prentice-Hall, Englewood Cliffs, NJ.

Kanopoulos, N. 1989. *Gallium Arsenide Digital Integrated Circuits: A Systems Perspective*. Prentice-Hall, Englewood Cliffs, NJ.

Leigh, B. 1993. Complex PLD & FPGA architectures, *ASIC & EDA*. (Feb.):44–50.

Matisof B. 1986. *Handbook of Electrostatic Controls (ESD)*. Van Nostrand Reinhold, New York.

Matthew, P. 1984. *Choosing and Using ECL*. McGraw-Hill, New York.

Rosenstark, S. 1994. *Transmission Lines in Computer Engineering*. McGraw-Hill, New York.

Scarlett, J.A. 1972. *Transistor–Transistor Logic and Its Interconnections*. Van Nostrand Reinhold, London.

Schilling, D. and Belove, C. 1989. *Electronic Circuits: Discrete and Integrated*, 3rd ed. McGraw-Hill, New York.

Texas Instruments, 1984. *High-Speed CMOS Logic Data Book*. Texas Instruments, Dallas, TX.

Xilinx, 1993. *The Programmable Gate Array Data Book*. Xilinx, Inc., San Jose, CA.

Further Information

For the practitioner, the following technical journals are excellent sources dealing with current trends in both discrete logic and programmable devices.

Electronic Design, published 30 times a year by Penton Publishing, Inc. of Cleveland, OH, covers a very broad range of design-related topics. It is free to qualified subscribers.

EDN, published by Cahners Publishing Company, Newton, MA, appears 38 times annually. It also deals with the entire range of electronics design topics and is free to qualified subscribers.

Computer Design, Pennwell Publishing, Nashua, NH, appears monthly. It focuses more specifically on digital design topics than the foregoing. Like them, it is free to qualified subscribers.

Integrated System Design, a monthly periodical of the Verecom Group, Los Altos, CA, is a valuable source of current information about programmable logic and the associated design tools.

© 2006 by Taylor & Francis Group, LLC

For the researcher, the *IEEE Journal of Solid State Circuits* is perhaps the best single source of information about advances in digital logic. The IEEE sponsors a number of conferences annually that deal in one way or another with digital logic families and programmable logic.
Manufacturer's databooks and application notes provide much current information about logic, especially with regard to applications information. They usually assume some background knowledge, however, and can be difficult to use as a first introduction. Recommended textbooks to provide that background include the following.

Brown, S.D., Francis, R., Rose, J., and Vranesic, Z. 1992. *Field-Programmable Gate Arrays*. Kluwer Academic, Boston, MA.
Buchanan, J. 1990. *CMOS/TTL Digital System Design*. McGraw-Hill, New York.
Jenkins, J.H. 1994. *Designing with FPGAs and CPLDs*. Prentice-Hall, Englewood Cliffs, NJ.
Matthew, P. 1984. *Choosing and Using ECL*. McGraw-Hill, New York.
Rosenstark, S. 1994. *Transmission Lines in Computer Engineering*. McGraw-Hill, New York.

© 2006 by Taylor & Francis Group, LLC

8

Memory Devices

8.1 Introduction 8-1
8.2 Memory Organization 8-2
Memory Hierarchy • System Level Memory Organization • Memory Device Organization
8.3 Memory Device Types 8-5
Read-Only Memory • Random Access Memory (RAM) • Special Memory Structures
8.4 Interfacing Memory Devices 8-15
Accessing DRAMs • Refreshing the DRAM
8.5 Error Detection and Correction 8-20

Shih-Lien Lu

8.1 Introduction

Memory is an essential part of any computation system. It is used to store both the computation instructions and the data. Logically, memory can be viewed as a collection of sequential locations, each with a unique address as its label and capable of storing information. Accessing memory is accomplished by supplying the address of the desired data to the device.

Memory devices can be categorized according to their functionality and fall into two major categories, **read-only-memory (ROM)**, and write-and-read memory or random-access memory (RAM). There is also another subcategory of ROM: mostly read but sometimes write memory or flash ROM memory. Within the RAM category there are two types of memory devices differentiated by storage characteristics, **static RAM (SRAM)** and **dynamic RAM (DRAM)** respectively. DRAM devices need to be refreshed periodically to prevent the corruption of their contents due to charge leakage. SRAM devices, on the other hand, do not need to be refreshed.

Both SRAM and DRAM are volatile memory devices, which means that their contents are lost if the power supply is removed from these devices. Nonvolatile memory, the opposite of volatile memory, retains its contents even when the supply power is turned off. All current ROM devices, including mostly read sometimes write devices are nonvolatile memories. Except for a very few special memories, these devices are all interfaced in a similar way. When an address is presented to a memory device, and sometimes after a control signal is strobed, the information stored at the specified address is retrieved after a certain delay. This process is called a **memory read**. This delay, defined as the time taken from address valid to data ready, is called **memory read access time**. Similarly, data can be stored into the memory device by performing a **memory write**. When writing, data and an address are presented to the memory device with the activation of a write control signal. There are also other control signals used to interface. For example, most of the memory devices in packaged chip format has a chip select (or chip enable) pin. Only when this pin is asserted, the particular memory device gets active. Once an address is supplied to the chip, internal address decoding logic is used to pinpoint the particular content for output. Because of the nature of the circuit structure used in implementing the decoding logic, a memory device usually needs to recover before a subsequent read or write can be performed. Therefore, the time between subsequent address issues is called **cycle time**. Cycle time is usually twice as long as the access time. There are other timing requirements for memory devices. These timing parameters play a very important role in interfacing the

© 2006 by Taylor & Francis Group, LLC

memory devices with computation processors. In many situations, a memory device's timing parameters affect the performance of the computation system greatly.

Some special memory structures do not follow the general accessing scheme of using an address. Two of the most frequently used are content addressable memory (CAM), and first-in-first-out (FIFO) memory. Another type of memory device, which accepts multiple addresses and produces several results at different ports, is called multiport memory. There is also a type of memory that can be written in parallel, but is read serially. It is referred to as video RAM or VDRAM since they are used primarily in graphic display applications. We will discuss these in more detail later.

8.2 Memory Organization

There are several aspects to memory organization. We will take a top down approach in discussing them.

Memory Hierarchy

The speed of memory devices has been lagging behind the speed of processors. As processors become faster and more capable, larger memory spaces are required to keep up with the every increasing software complexity written for these machines. Figure 8.1(a) illustrates the well-known Moore's law, depicting the exponential growth in central processing unit (CPU) and memory capacity. Although CPUs' speed continues to grow with the advancement of technology and design technique (in particular pipelining), due to the nature of increasing memory size, more time is needed to decode wider and wider addresses and to sense the information stored in the ever-shrinking storage element. The speed gap between CPU and memory devices continues to grow wider. Figure 8.1(b) illustrates this phenomenon.

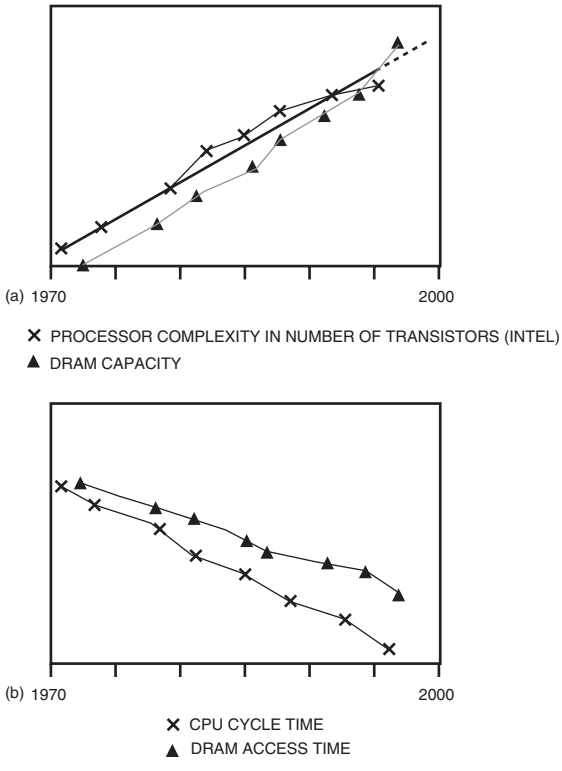


FIGURE 8.1 (a) Processor and memory development trend, (b) speed difference between RAM and CPU.

© 2006 by Taylor & Francis Group, LLC

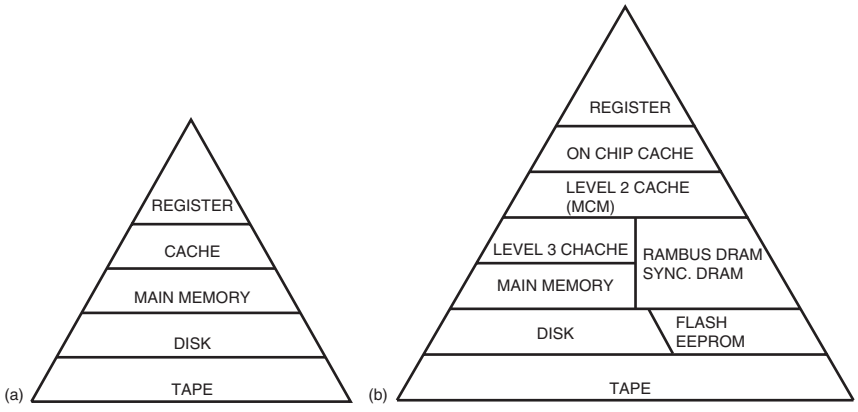


FIGURE 8.2 Traditional memory hierarchy.

The strategy used to remedy this problem is called **memory hierarchy**. Memory hierarchy works because of the locality property of memory references due to the sequentially fetched program instructions and the conjugation of related data. In a hierarchical memory system there are many levels of memory hierarchies. A small amount of very fast memory is usually allocated and brought right next to the central processing unit to help match up the speed of the CPU and memory. As the distance becomes greater between the CPU and memory, the performance requirement for the memory is relaxed. At the same time, the size of the memory grows larger to accommodate the overall memory size requirement. Some of the memory hierarchies are *registers, cache, main memory,* and *disk*. Figure 8.2 illustrates the general memory hierarchy employed in a traditional system. When a memory reference is made, the processor accesses the memory at the top of the hierarchy. If the desired data is in the higher hierarchy, a "bit" is encountered and information is obtained quickly. Otherwise a *miss* is encountered. The requested information must be brought up from a lower level in the hierarchy. Usually memory space is divided into blocks so that it can be transferred between levels in groups. At the cache level a chunk is called a *cache block* or a *cache line*. At the main memory level a chunk is referred to as a memory *page*. A miss in the cache is called a *cache miss* and a miss in the main memory is called a *page fault*. When a miss occurs, the whole block of memory containing the requested missing information is brought in from the lower hierarchy as mentioned before. If the current memory hierarchy level is full when a miss occurs, some existing blocks or pages must be removed and sometimes written back to a lower level to allow the new one(s) to be brought in. There are several different *replacement algorithms*. One of the most commonly used is the *least recently used* (LRU) replacement algorithm.

In modern computing systems, there may be several sublevels of cache within the hierarchy of cache. The general principle of memory hierarchy is that the farther away from the CPU it is, the larger its size, slower its speed, and the cheaper its price per memory unit becomes. Because the memory space addressable by the CPU is normally larger than necessary for a particular software program at a given time, disks are used to provide an economical supplement to main memory. This technique is called *virtual memory*. Besides disks there are tapes, optical drives, and other backup devices, which we normally call *backup storage*. They are used mainly to store information that is no longer in use, to protect against main memory and disk failures, or to transfer data between machines.

System Level Memory Organization

On the system level, we must organize the memory to accomplish different tasks and to satisfy the need of the program. In a computation system, addresses are supplied by the CPU to access the data or instruction. With a given address width a fixed number of memory locations may be accessed. This is referred to as the *memory address space*. Some processing systems have the ability to access another separate space called *input/output* (I/O) *address space*. Others use part of the memory space for I/O purposes.

© 2006 by Taylor & Francis Group, LLC

This style of performing I/O functions is called *memory–mapped I/O*. The memory address space defines the maximum size of the directly addressable memory that a computation system can access using memory type instructions. For example, a processor with address width of 16-b can access up to 64 K different locations (memory entries), whereas a 32-b address width can access up to 4 Gig different locations. However, sometimes we can use indirection to increase the address space. The method used by 80X86 processors provides an excellent example of how this is done. The address used by the user or programmer in specifying a data item stored in the memory system is called a *logical address*. The address space accessed by the logical address is named *logical address space*. However, this logical address may not necessarily be used directly to index the physical memory. We called the memory space accessed by physical address the *physical address space*. When the logical space is larger than the physical space, then memory hierarchy is required to accommodate the difference of space sizes and store them in a lower hierarchy. In most of the current computing systems, a hard-disk is used as this lower hierarchy memory. This is termed *virtual memory system*. This mapping of logical address to physical address could be either linear or nonlinear. The actual address calculation to accomplish the mapping process is done by the CPU and the memory management unit (MMU). Thus far, we have not specified the exact size of a memory entry. A commonly used memory entry size is one byte. For historical reasons, memory is organized in bytes. A byte is usually the smallest unit of information transferred with each memory access. Wider memory entries are becoming more popular as the CPU continues to grow in speed and complexity. There are many modern systems that have a data width wider than a byte. A common size is a double word (32 b). in current desktop. As a result, memory in bytes is organized in sections of multibytes. However, due to need for backward compatibility, these wide datapath systems are also organized to be byte addressable. The maximum width of the memory transfer is usually called *memory word length*, and the size of the memory in bytes is called *memory capacity*. Since there are different memory device sizes, the memory system can be populated with different sized memory devices.

Memory Device Organization

Physically, within a memory device, cells are arranged in a two-dimensional array with each of the cell capable of storing 1 b of information. This matrix of cells is accessed by specifying the desired row and column addresses. The individual row enable line is generated using an address decoder while the column

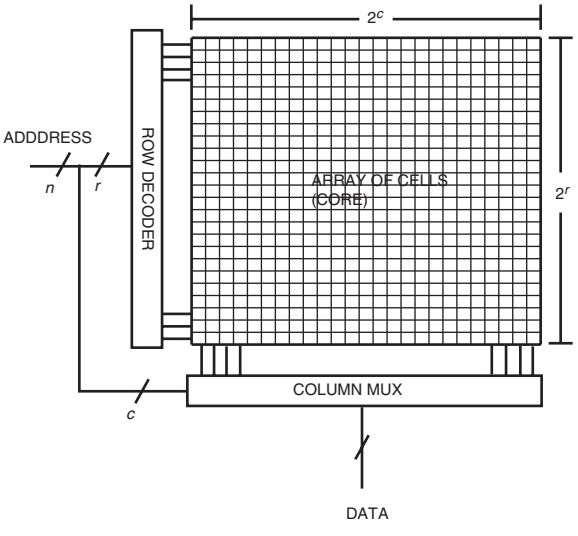


FIGURE 8.3 Generic memory structure.

© 2006 by Taylor & Francis Group, LLC

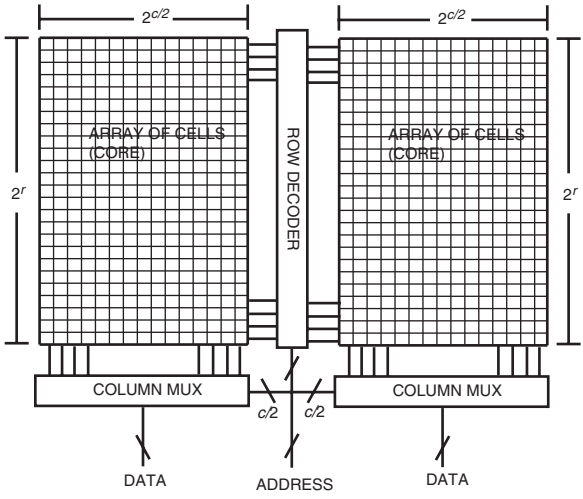


FIGURE 8.4 Divided memory structure.

is selected through a multiplexer. There is usually a sense amplifier between the column bit line and the multiplexer input to detect the content of the memory cell it is accessing. Figure 8.3 illustrates this general memory cell array described with r bit of row address and c bit of column address. With the total number of $r + c$ address bits, this memory structure contains 2^{r+c} number of bits. As the size of memory array increases, the row enable lines, as well as the column bit lines, become longer. To reduce the capacitive load of a long row enable line, the row decoders, sense amplifiers, and column multiplexers are often placed in the middle of divided matrices of cells, as illustrated in Fig. 8.4. By designing the multiplexer differently we are able to construct memory with different output width, for example, $\times 1$, $\times 8$, $\times 16$, etc. In fact, memory designers go to great efforts to design the column multiplexers so that most of the fabrication masks may be shared for memory devices that have the same capacity but with different configurations.

8.3 Memory Device Types

As mentioned before, according to the functionality and characteristics of memory, we may divide memory devices into two major categories: ROM and RAM. We will describe these different type of devices in the following sections.

Read-Only Memory

In many systems, it is desirable to have the system level software (for example, basic input/output system [BIOS]) stored in a read-only format, because these types of programs are seldom changed. Many embedded systems also use read-only memory to store their software routines because these programs also are never changed during their lifetime, in general. Information stored in the read-only memory is permanent. It is retained even if the power supply is turned off. The memory can be read out reliably by a simple current sensing circuit without worrying about destroying the stored data. Figure 8.5 shows the general structure of a read-only memory (ROM). The effective switch position at the intersection of the word-line/bit-line determines the stored value. This switch could be implemented using different technologies resulting in different types of ROMs.

Masked Read-Only Memory (ROM)

The most basic type of this read-only-memory is called masked ROM, or simply ROM. It is programmed at manufacturing time using fabrication processing masks. ROM can be produced using different

© 2006 by Taylor & Francis Group, LLC

technologies, bipolar, complementary metal oxide semiconductor (CMOS), n-channel metal oxide semiconductor (nMOS), p-channel metal oxide semiconductor (pMOS), etc. Once they are programmed there are no means to change their contents. Moreover, the programming process is performed at the factory.

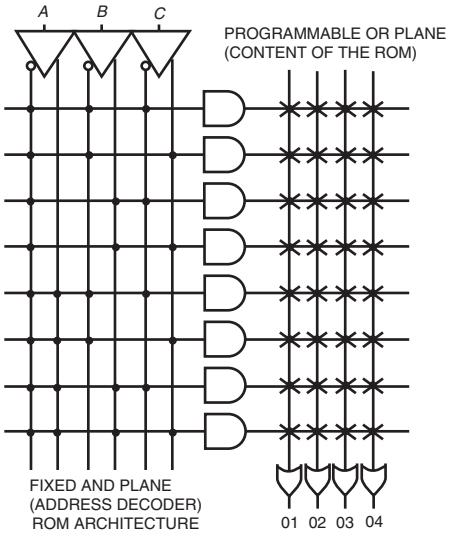


FIGURE 8.5 General structure of a ROM (an 8 × 4 ROM).

Programmable Read-Only Memory (PROM)

Some read-only memory is one-time programmable, but it is programmable by the user at the user's own site. This is called programmable read-only memory (PROM). It is also often referred to as write once memory (WOM). PROMs are based mostly on bipolar technology, since this technology supports it very well. Each of the single transistors in a cell has a fuse connected to its emitter. This transistor and fuse make up the memory cell. When a fuse is blown, no connection can be established when the cell is selected using the row line. Thereby a zero is stored. Otherwise, with the fuse intact, a logic one is represented. The programming is done through a programmer called a PROM programmer or PROM burner. Figure 8.6 illustrates the structure of a bipolar PROM cell and its cross section when fabricated.

Erasable Programmable Read-Only Memory (EPROM)

It is sometimes inconvenient to program the ROM only once. Thus, the erasable PROM is designed. This type of erasable PROM is called EPROM. The programming of a cell is achieved by avalanche injection of high-energy electrons from the substrate through the oxide. This is accomplished by applying a high drain voltage, causing the electrons to gain enough energy to jump over the 3.2-eV barrier between the substrate and silicon dioxide thus collecting charge at the floating gate. Once the applied voltage is removed, this charge is trapped on the floating gate. Erasing is done using an ultraviolet (UV) light eraser. Incoming

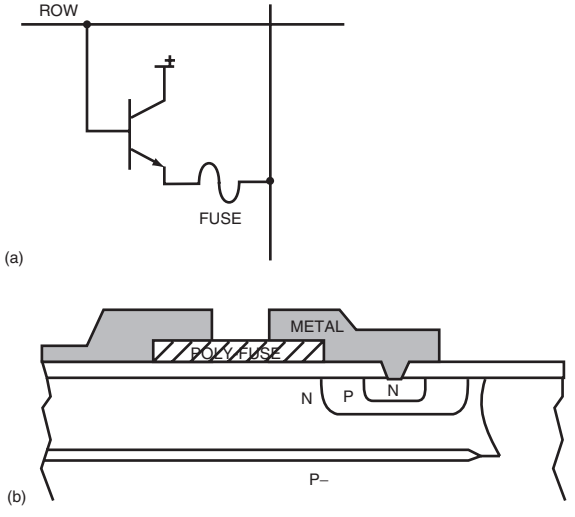


FIGURE 8.6 Bipolar PROM: (a) bipolar PROM cell, (b) cross section of a bipolar PROM cell.

© 2006 by Taylor & Francis Group, LLC

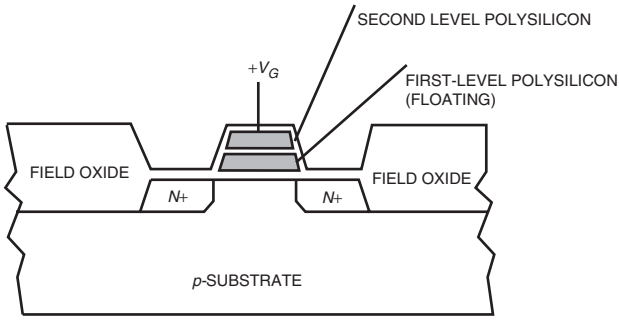


FIGURE 8.7 Cross section of a floating gate EPROM cell.

UV light increases the energy of electrons trapped on the floating gate. Once the energy is increased above the 3.2-eV barrier, the electrons leave the floating gate and move toward the substrate and the selected gate. Therefore, these EPROM chips all have windows on their packages where erasing UV light can reach inside the packages to erase the content of cells. The erase time is usually in minutes. The presence of a charge on the floating gate will cause the MOS transistor to have a high threshold voltage. Thus, even with a positive select gate voltage applied at the second level of polysilicon the MOS remains to be turned off. The absence of a charge on the floating gate causes the MOS to have a lower threshold voltage. When the gate is selected the transistor will turn on and give the opposite data bit. Figure 8.7 illustrates the cross section of a EPROM cell with floating gate. EPROM technologies that migrate toward smaller geometries make floating-gate discharge (erase) via UV light exposure increasingly difficult. One problem is that the width of metal bit lines cannot reduce proportionally with advancing process technologies. EPROM metal width requirements limit bit-lines spacing, thus reducing the amount of high-energy photons that reach charged cells. Therefore, EPROM products built on submicron technologies will face longer and longer UV exposure time.

Electrical Erasable Read-Only Memory (EEPROM)

Reprogrammability is a very desirable property. However, it is very inconvenient to use a separate light-source eraser for altering the contents of the memory. Furthermore, even a few minutes of erase time is intolerable. For this reason, an erasable PROM was designed called electrical erasable PROM (EEPROM). EEPROM permits new applications where erasing is done without removing the device from the system it resides in. There are a few basic technologies used in the processing of EEPROMs or electrical reprogrammable ROMs. All of them uses the Fowler-Nordheim tunneling effect to some extent. In this tunneling effect, cold electrons jump through the energy barrier at a silicon-silicon dioxide interface and into the oxide conduction band. This can only happen when the oxide thickness is of 100 Å or less, depending on the technology. This tunneling effect is reversible, allowing the reprogrammable ROMs to be used over and over again. One of the first electrical erasable PROMs is the electrical alterable ROM (EAROM) which is based on metal-nitrite-oxide silicon (MNOS) technology. The other is EEPROM, which is based on silicon floating gate technology used in fabricating EPROMs. The floating gate type of EEPROM is favored because of its reliability and density. The major difference between EPROM and EEPROM is in the way they discharge the charge stored in the floating gate. EEPROM must discharge floating gates electrically as opposed to using an UV light source in an EPROM device where electrons absorb photons from the UV radiation and gain enough energy to jump the silicon/silicon-dioxide energy barrier in the reverse direction as they return to the substrate. The solution for the EEPROM is to pass low-energy electrons through the thin oxide through high field (10^7 V/cm^2). This is known as the Fowler-Nordheim tunneling, where electrons can pass a short distance through the forbidden gap of the insulator and enter the conduction bank when the field applied is high enough. There are three common types of flash EEPROM cells. One uses the erase gate (three levels of polysilicon), the second and third use source and drain, respectively, to

© 2006 by Taylor & Francis Group, LLC

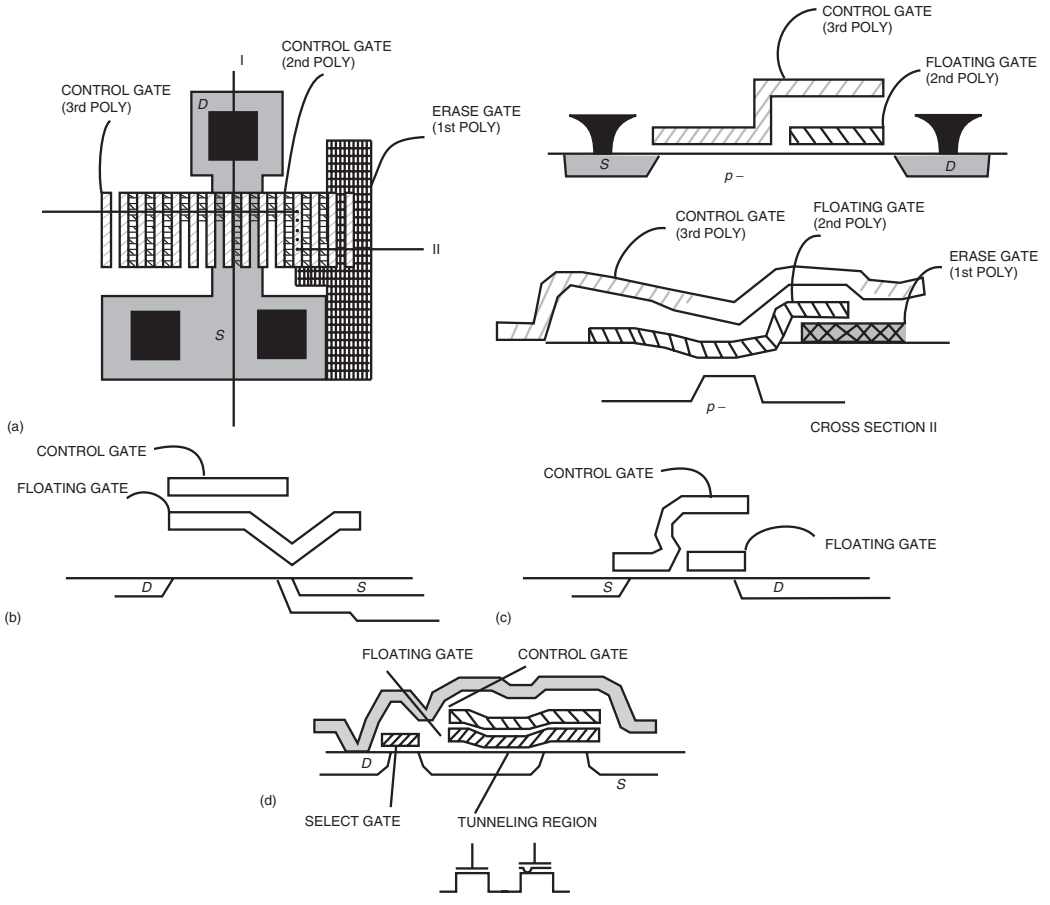


FIGURE 8.8 (a) Triple poly EEPROM cell layout and structure, (b) flotox EEPROM cell structure (source programming), (c) EEPROM with drain programming, (d) another source programming EEPROM.

erase. Figures 8.8(a)–8.8(d) illustrate the cross sections of different EEPROMs. To realize a small EEPROM memory cell, the NAND structure was proposed in 1987. In this structure, cells are arranged in series. By using different patterns, an individual cell can be detected whether it is programmed or not. From the user's point of view, EEPROMs differs from RAM only in their write time and number of writes allowed before failure occurs. Early EEPROMs were hard to use because they have no latches for data and address to hold values during the long write operations. They also require a higher programming voltage, other than the operating voltage. Newer EEPROMs use charge pumps to generate the high programming voltage on the chip so the user does not need to provide a separate programming voltage.

Flash-EEPROM

This type of erasable PROM lacks the circuitry to erase individual locations. When you erase them, they are erased completely. By doing so, many transistors may be saved, and larger memory capacities are possible. Note that sometimes you do not need to erase before writing. You can also write to an erased, but unwritten location, which results in an average write time comparable to an EEPROM. Another important thing to know is that writing zeros into a location charges each of the flash EEPROM's memory cells to the same electrical potential so that subsequent erasure drains an equal amount of free charge (electrons) from each cell. Failure to equalize the charge in each cell prior to erasure can result in the overerasure of some cells by dislodging bound electrons in the floating gate and driving them out. When a floating gate is depleted in this way, the corresponding transistor can never be turned off again, thus destroying the flash EEPROM.

© 2006 by Taylor & Francis Group, LLC

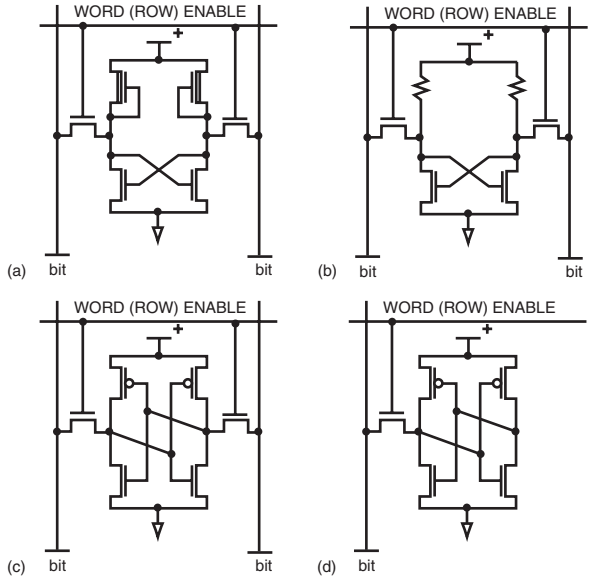


FIGURE 8.9 Different SRAM cells: (a) six-transistor SRAM cell with depletion transistor load, (b) four-transistor SRAM cell with polyresistor load, (c) CMOS six-transistor SRAM cell, (d) five-transistor SRAM cell.

Random Access Memory (RAM)

RAM stands for random-access memory. It is really read-write memory because ROM is also random access in the sense that given a random address the corresponding entry is read. RAM can be categorized by content duration. A static RAM's contents is always retained, as long as power is applied. A DRAM, on the other hand, needs to be refreshed every few milliseconds. Most RAMs by themselves are volatile, which means that without the power supply their content will be lost. All of the ROMs mentioned in the previous section are nonvolatile. RAM can be made nonvolatile by using a backup battery.

Static Random Access Memory (SRAM)

Figure 8.9 shows various SRAM memory cells (6T, 5T, 4T). The six transistor (6T) SRAM cell is commonly used SRAM. The crossed coupled inverters in a SRAM cell retain the information indefinitely, as long as the power supply is on since one of the pull-up transistors supplies current to compensate for the leakage current. During a read, bit and bitbar line are precharged while the word enable line is held low. Depending on the content of the cell, one of the lines is discharged slightly causing the precharged voltage to drop when the word enable line is strobed. This difference in voltage between the bit and bitbar lines is sensed by the sense amplifier, which produces the read result. During a write process, one of the bit/bitbar lines is discharged, and by strobing the word enable line the desired data is forced into the cell before the word line goes away.

Figure 8.10 gives the circuit of a complete SRAM circuit design with only one column and one row shown. One of the key design parameters of a SRAM cell is to determine the size of transistors used in the memory cell. We first need to determine the criteria used in sizing transistors in a CMOS 6-transistor/cell SRAM. There are three transistor sizes to choose in a 6-transistor CMOS SRAM cell due to symmetry. They are the pMOS pull-up size, the nMOS pull-down size, and the nMOS access transistor (also called the pass-transistor gate) size. There are two identical copies of each transistor, giving a total of six. Since the sole purpose of the pMOS pull-up is to supply enough current in overcoming junction current leakage, we should decide this size first. This is also the reason why some SRAMs completely remove the two pMOS transistors and replace them with two 10-GΩ polysilicon resistors giving the 4T cell shown in Fig. 8.9. Since one of the goals is to make your cell layout as small as possible, pMOS pull-up is chosen to be minimum

© 2006 by Taylor & Francis Group, LLC

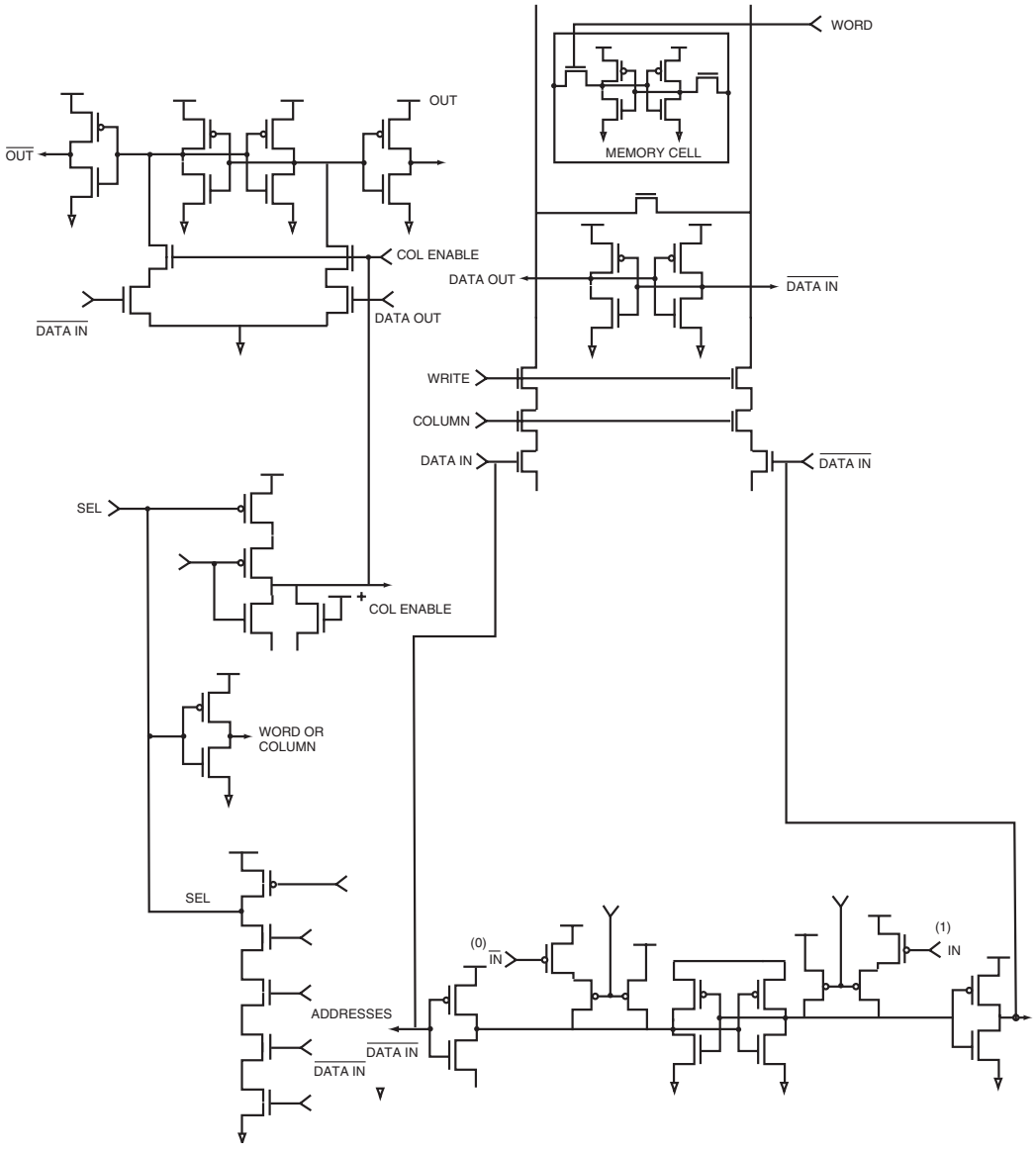


FIGURE 8.10 A complete circuit of an SRAM memory.

both in its length and width. Only when there is room available (i.e., if it does not increase the cell layout area), the length of pMOS pull-up is increased. By increasing the length of pMOS pull-up transistors, the capacitance on the crossed-coupled inverters output nodes is increased. This helps in protecting against certain soft errors. It also makes the cell slightly easier to write.

The next step is to choose the nMOS access transistor size. This is a rather complicated process. To begin we need to determine the length of this transistor. It is difficult to choose because, on one hand, we want it also to be minimum in order to reduce the cell layout size. However, on the other hand, a column of n SRAM bits (rows) has to have n of access transistors connected to the bit or bitbar line. If each of the cells leaks just a small amount of current, the leakage is multiplied by n. Thus, the bit or bitbar line, which one might think should be sitting at the bit-line pull-up (or precharge) voltage, is actually pulled down by this leakage. Thus, the bit or bitbar line high level is lower than the intended voltage. When this happens, the voltage difference between the bit and bitbar lines, which is seen by the sense amplifier

© 2006 by Taylor & Francis Group, LLC

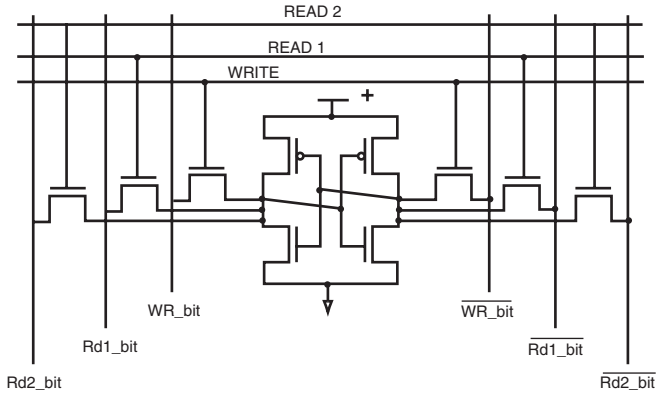


FIGURE 8.11 Multiported CMOS SRAM cell (shown with 2-read and 1-write).

during a read, is smaller than expected, perhaps catastrophically so. Thus, if the transistors used are not particularly leaky or n is small, a minimum sized nMOS is sufficient. Otherwise, a larger sized transistor should be used. Beside considering leakage, there are three other factors that may affect the transistor sizes of the two nMOSs. They are: (1) cell stability, (2) speed, and (3) layout area. The first factor, cell stability, is a DC phenomenon. It is a measure of the cell's ability to retain its data when reading and to change its data when writing. A read is done by creating a voltage difference between the bit and bitbar lines (which are normally precharged) for the sense amplifier to differentiate. A write is done by pulling one of the bit or bitbar lines down completely. Thus, one must design the size to satisfy the cell stability while achieving the maximum read and write speed and maintaining the minimum layout area.

Much work has been done in writing computer-aided design (CAD) tools that automatically size transistors for arrays of SRAM cells, and then do the polygon layout. Generally, these are known as SRAM macrogenerators or RAM compilers. These generated SRAM blocks are used as drop ins in many application specific intergrated circuits (ASICs). Standard SRAM chips are also available in many different organizations. Common ones are arranged in 4 b, bytes, and double words (32 b) in width.

There is also a special type of SRAM cell used in computers to implement registers. These are called multiple port memories. In general, the contents can be read by many different requests at the same time. Figure 8.11 shows a dual-read port single-write port SRAM cell. When laying out SRAM cells, adjacent cells usually are mirrored to allow sharing of supply or ground lines. Figure 8.12 illustrates the layout of four adjacent SRAM cells using a generic digital process design rules. This block of four cells can be repeated in a two-dimensional array format to form the memory core.

Direct Random Access Memory (DRAM)

The main disadvantage of SRAM is in its size since it takes six transistors (or at least four transistors and two resistors) to construct a single memory cell. Thus, the DRAM is used to improve the capacity. There are different DRAM cell designs. There is the four-transistor DRAM cell, three-transistor DRAM cell, and the one-transistor DRAM cell. Figures 8.13 shows the corresponding circuits for these cells. Data writing is accomplished in a three-transistor cell by keeping the RD line low (see Fig. 8.13(b)) while strobing the WR line with the desired data to be written is kept on the bus. If a 1 is desired to be stored, the gate of T2 is charged turning on T2. This charge remains on the gate of T2 for a while before the leakage current discharges it to a point where it cannot be used to turn on T2. When the charge is still there, a read can be performed by precharging the bus and strobing the RD line. If a 1 is stored, then both T2 and T3 are on during a read, causing the charge on the bus to be discharged. The lowering of voltage can be picked up by the sense amplifier. If a zero is stored, then there is no direct path from the bus to ground, thus the charge on the bus remains. To further reduce the area of a memory cell, a single transistor cell is often used. Figure 8.13(c) shows the single transistor cell with a capacitor. Usually, two columns of cells

© 2006 by Taylor & Francis Group, LLC

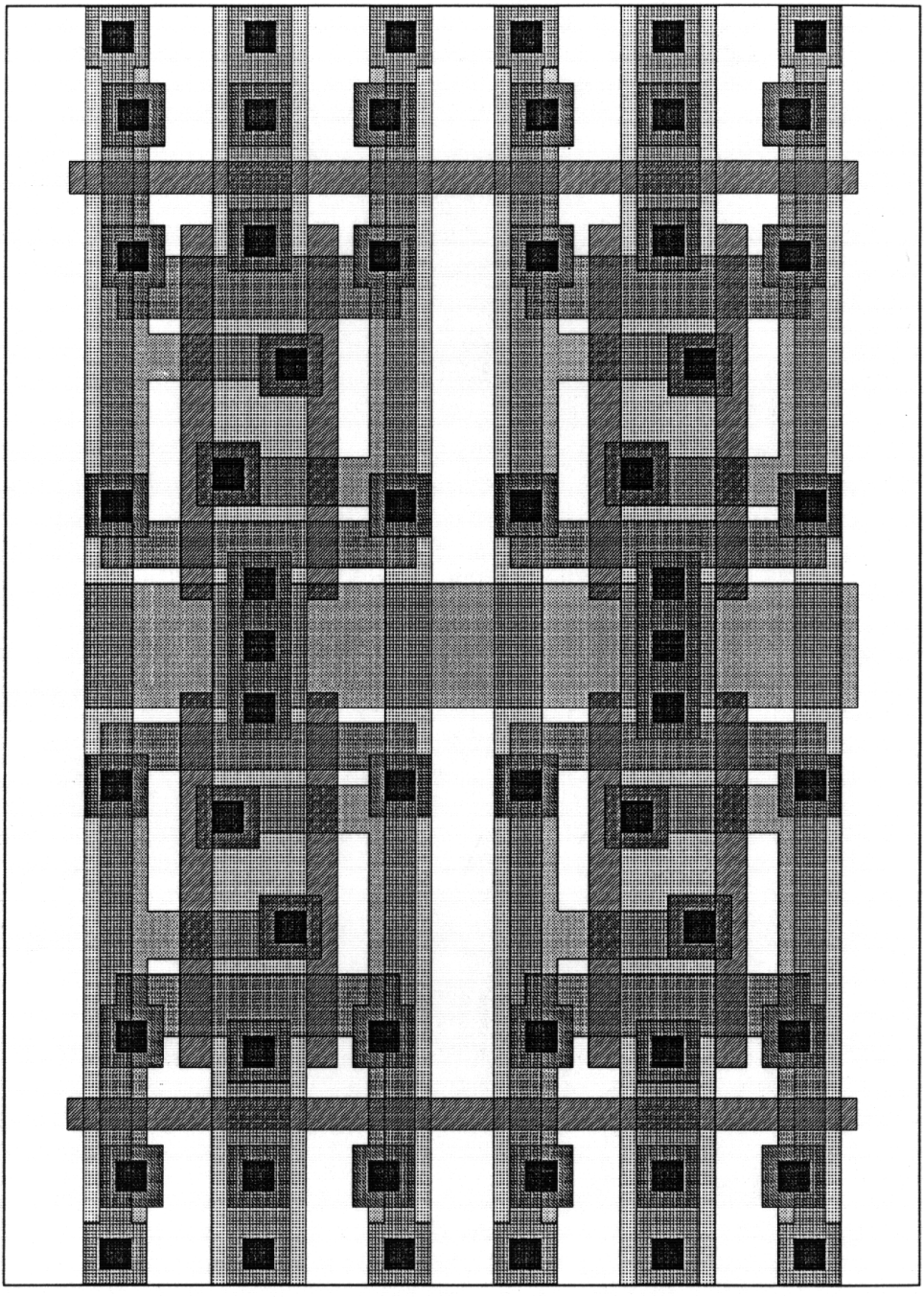


FIGURE 8.12 Layout example of four abutted 6–t SRAM cells.

© 2006 by Taylor & Francis Group, LLC

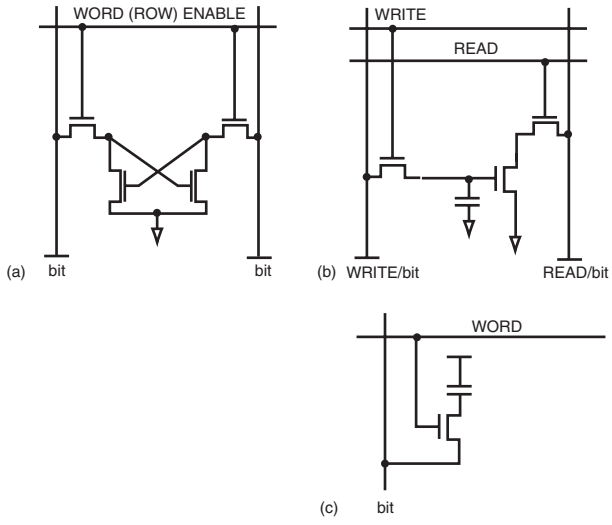


FIGURE 8.13 Different DRAM cells: (a) four-transistor DRAM cell, (b) three-transistor DRAM cell, (c) one-transistor DRAM cell.

are mirror images of each other to reduce the layout area. The sense amplifier is shared, as shown in Fig. 8.14. In this one-transistor DRAM cell, there is a capacitor used to store the charge, which determines the content of the memory. The amount of the charge in the capacitor also determines the overall performance of the memory. A continuing goal is to downscale the physical area of this capacitor to achieve higher and higher density. Usually, as one reduces the area of the capacitor, the capacitance also decreases. One approach is to increase the surface area of the storage electrode without increasing the layout area by employing stacked capacitor structures, such as finned or cylindrical structures. Certain techniques can be used to utilize a cylindrical capacitor structure with hemispherical grains. Figure 8.15 illustrates the cross section of a one-transistor DRAM cell with the cylindrical capacitor structure. Since the capacitor is charged by a source follower of the pass transistor, these capacitors can be charged maximally to a threshold voltage drop from the supply voltage. This reduces the total charge stored and affects performance, noise margin, and density. Frequently, to avoid this problem, the word lines are driven above the supply voltage when the data are written. Figure 8.16 shows typical layout of one-transistor DRAM cells. The writing is done by putting either a 0 or 1 (the desired data to store) on the read/writing line. Then the row select line is strobed. A zero or one is stored in the capacitor as charge. A read is performed with precharging

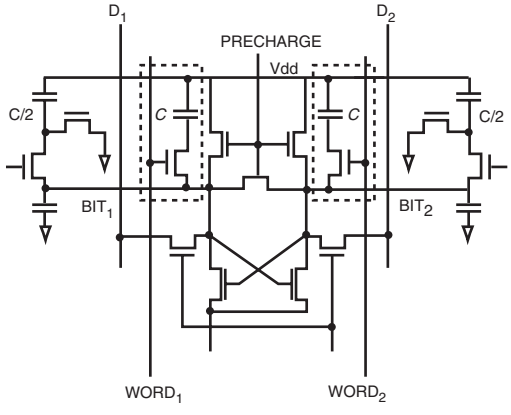


FIGURE 8.14 DRAM sense amplifier with 2 bit lines and 2 cells.

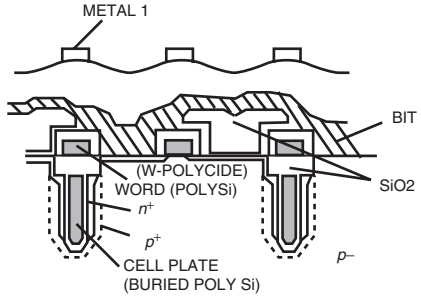


FIGURE 8.15 Cross section view of trenched DRAM cells.

© 2006 by Taylor & Francis Group, LLC

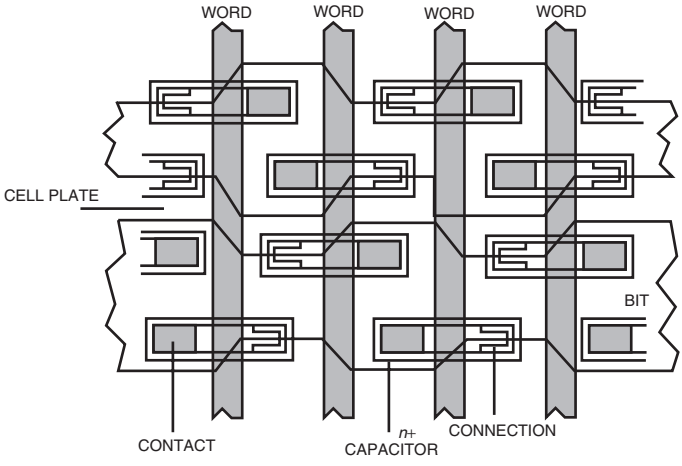


FIGURE 8.16 Physical layout of trenched DRAM cells.

the read/write line then strobing the row select. If a zero is stored due to charge sharing, the voltage on the read/write line decreases. Otherwise, the voltage remains. A sense amplifier is placed at the end to pick up if there is a voltage change or not. DRAM differs from SRAM in another aspect. As the density of DRAM increases, the amount of charge stored in a cell reduces. It becomes more subject to noise. One type of noise is caused by radiation called alpha particles. These particles are helium nuclei that are present in the environment naturally or emitted from the package that houses the DRAM die. If an alpha particle hits a storage cell, it may change the state of the memory. Since alpha particles can be reduced, but not eliminated, some DRAMs institute error detection and correction techniques to increase their reliability. Another difference between DRAMs and SRAMs is in the number of address pins needed for a given size RAM. SRAM chips require all address bits to be given at the same time. DRAMs, however, utilize time-multiplex address lines. Only half of the address bits are given at a given time. They are divided by rows and columns. An extra control signal is thus required. This is the reason why DRAM chips have two address strobe signals: row address strobe (RAS) and column address strobe (CAS).

Special Memory Structures

The trend in memory devices is toward larger, faster and better-performance products. There is a complementary trend toward the development of special purpose memory devices. Several types of special purpose RAMs are offered for particular applications such as content addressable memory for cache memory, line buffers (FIFOs) for office automation machines, frame buffers for TV and broadcast equipment, and graphics buffers for computers.

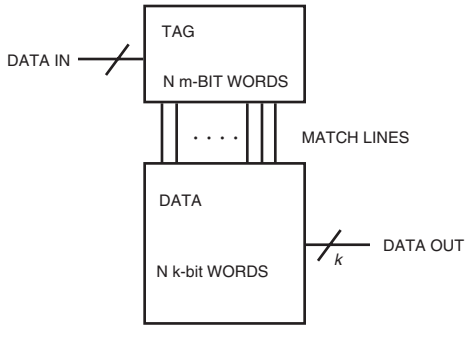


FIGURE 8.17 Functional view of a CAM.

Content Addressable Memory (CAM)

A special type of memory called content addressable memory (CAM) or associative memory is used in many applications such as cache memory and associative processor. A CAM stores a data item consisting of a tag and a value. Instead of giving an address, a data pattern is given to the tag section of the CAM. This data pattern is matched with the content of the tag section. If an item in the tag section of the CAM matches the supplied data pattern, the CAM outputs the value associated with the matched tag. Figure 8.17 illustrates the basic structure of a CAM. CAM cells must be both readable and writable just like the RAM cell. Figure 8.18 shows a circuit diagram for a basic CAM cell with a match output signal. This output signal may be used as an input for some logic to determine the matching process.

© 2006 by Taylor & Francis Group, LLC

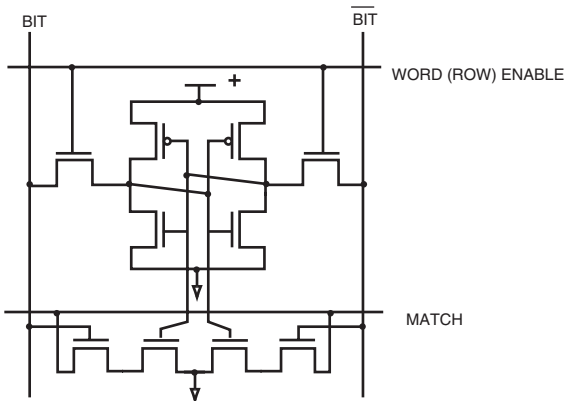


FIGURE 8.18 Static CMOS CAM cell.

First-In--First-Out/Queue (FIFO/Queue)

A FIFO/queue is used to hold data while waiting. It serves as the buffering region for two systems, which may have different rates of consuming and producing data. FIFO can be implemented using shift registers or RAMs with pointers.

Video RAM: Frame Buffer

There is a rapid growth in computer graphic applications. The technology that is most successful is termed raster scanning. In a raster scanning display system, an image is constructed with series of horizontal lines. Each of these lines are connected pixels of the picture image. Each pixel is represented with bits controlling the intensity. Usually there are three planes corresponding to each of the primary colors: red, green, and blue. These three planes of bit maps are called frame buffer or image memory. Frame buffer architecture affects the performance of a raster scanning graphic system greatly. Since these frame buffers needs to be read out serially to display the image line by line, a special type of DRAM memory called video memory is used. Usually these memory are dual ported with a parallel random access port for writing and a serial port for reading.

8.4 Interfacing Memory Devices

Besides capacity and type of devices, other characteristics of memory devices include its speed and the method of access. We mentioned in the Introduction memory access time. It is defined as the time between the address available to the divide and the data available at the pin for access. Sometimes the access time is measured from a particular control signal. For example, the time between the read control line ready to data ready is called the read command access time. The memory cycle time is the minimum time between two consecutive accesses. The memory write command time is measured from the write control ready to data stored in the memory. The memory latency time is the interval between the CPU issuing an address to data available for processing. The memory bandwidth is the maximum amount of memory capacity being transferred in a given time. Access is done with address, read/write control lines, and data lines. SRAM and ROMs are accessed similarly during read. Figure 8.19 shows the timing diagram of two SRAM read cycles. In both methods read cycle time is defined as the time period between consecutive read addresses. In the first method, SRAM acts as an asynchronous circuit. Given an address, the output of the SRAM changes and become valid after a certain delay, which is the read access time. The second method uses two control signals, chip select and output enable, to initiate the read process. The main difference is in the data output valid time. With the second method data output is only valid after the output enable signal is asserted, which allows several devices to be connected to the data bus. Writing SRAM and electrically

© 2006 by Taylor & Francis Group, LLC

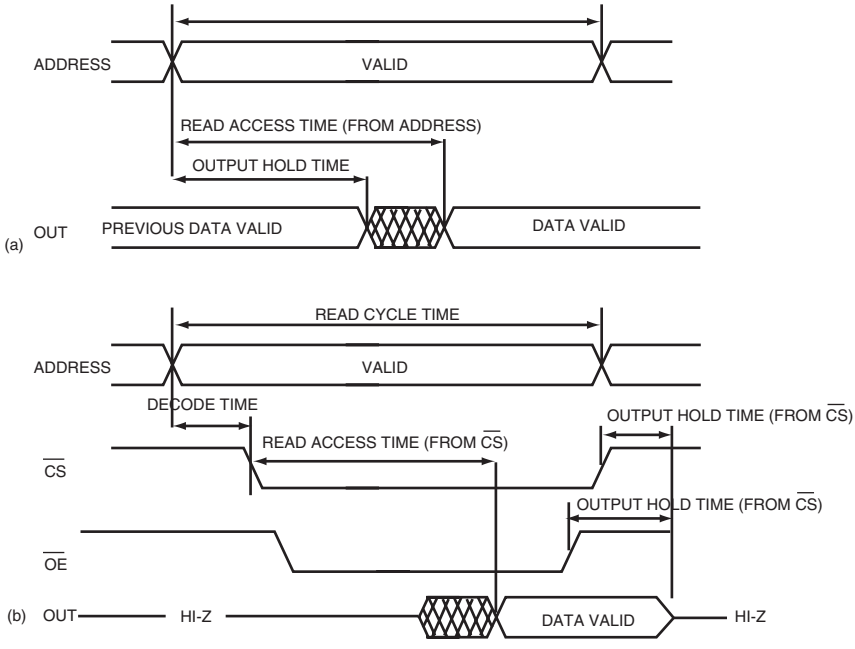


FIGURE 8.19 SRAM read cycle: (a) simple read cycle of SRAM (OE∼, CE∼ are all asserted and WE∼ is low), (b) SRAM read cycle.

reprogrammable ROM is slightly different. Since there are many different programmable ROMs and their writing processes depend on the technology used, we will not discuss the writing of ROMs here. Figure 8.20 shows the timing diagram of writing typical SRAM chips. Figure 8.20(a) shows the write cycle using the write enable signal as the control signal, whereas Fig. 8.20(b) shows the write cycle using chip enable signals. Accessing DRAM is very different from SRAM and ROMs. We will discuss the different access modes of DRAMs in the following section.

Accessing DRAMs

DRAM is very different from SRAM in that its row and column address are time multiplexed. This is done to reduce the pins of the chip package. Because of time multiplexing there are two address strobe lines for DRAM address, RAS and CAS. There are many ways to access the DRAM. We list the five most common ways.

Normal Read/Write

When reading, a row address is given first, followed by the row address strobe signal RAS. RAS is used to latch the row address on chip. After RAS, a column address is given followed by the column address strobe CAS. After certain delay (read access time) valid data appear on the data lines. Memory write is done similarly to memory read with only the read/write control signal reversed. There are three cycles available to write a DRAM. They are early write, read-modify-write, and late write cycles. Figure 8.21 shows only the early write cycle of a DRAM chip. Other write cycles can be found in most of the DRAM databooks.

Fast Page Mode (FPM) or Page Mode

In page mode (or fast page mode), a read is done by lowering the RAS when the row address is ready. Then, repeatedly give the column address and CAS whenever a new one is ready without cycling the RAS line. In this way a whole row of the two-dimensional array (matrix) can be accessed with only one RAS and the same row address. Figure 8.22 illustrates the read timing cycle of a page mode DRAM chip.

© 2006 by Taylor & Francis Group, LLC

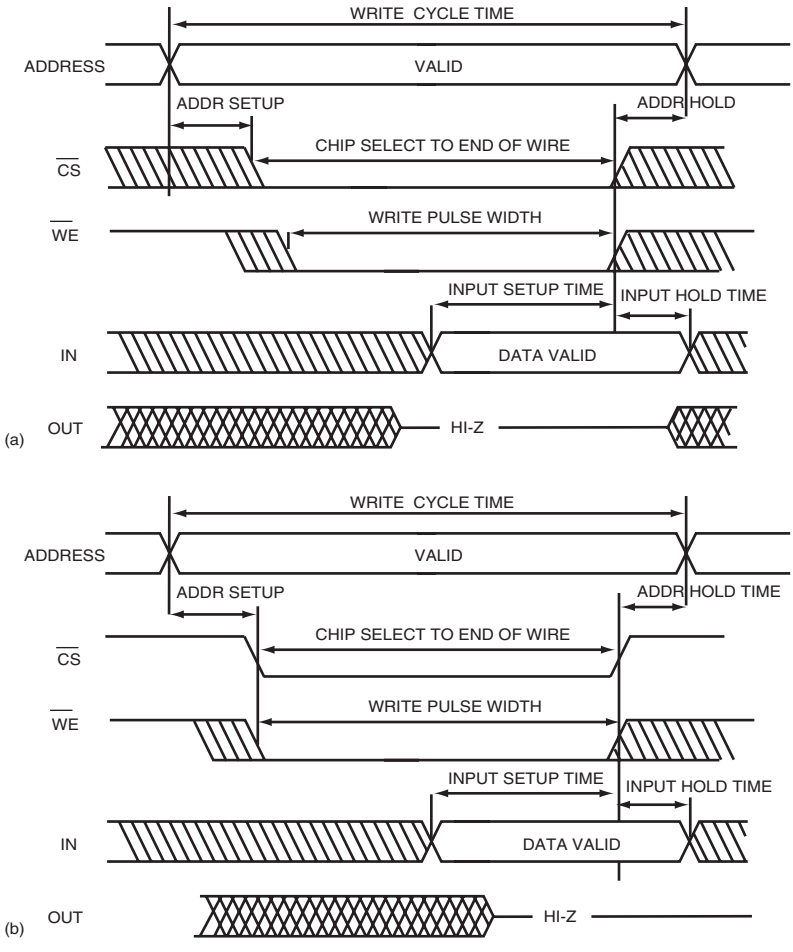


FIGURE 8.20 SRAM write cycles: (a) write enable controlled (b) chip enable controlled.

Static Column

Static column is almost the same as page mode except the CAS signal is not cycled when a new column address is given—thus the static column name.

Extended Date Output (EDO) Mode

In page mode, CAS must stay low until valid data reach the output. Once the CAS assertion is removed, data are disabled and the output pins goes to open circuit. With EDO DRAM, an extra latch following the sense amplifier allows the CAS line to return to high much sooner, permitting the memory to start precharging earlier to prepare for the next access. Moreover, data are not disabled after CAS goes high. With burst EDO DRAM, not only does the CAS line return to high, it can also be toggled to step though the sequence in burst counter mode, providing even faster data transfer between memory and the host. Figure 8.23 shows a read cycle of an EDO page mode DRAM chip. EDO mode is also called hyper page mode (HPM) by some manufactures.

Nibble Mode

In nibble mode after one CAS with a given column three more accesses are performed automatically without giving another column address (the address is assumed to be incremented from the given address).

© 2006 by Taylor & Francis Group, LLC

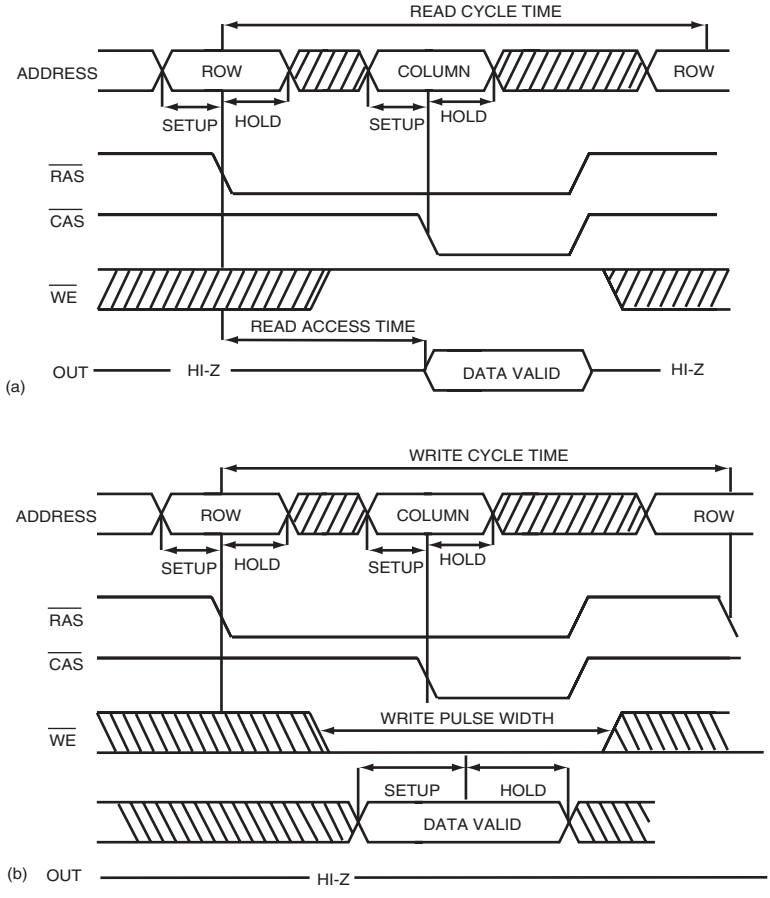


FIGURE 8.21 DRAM read and write cycles: (a) read, (b) write.

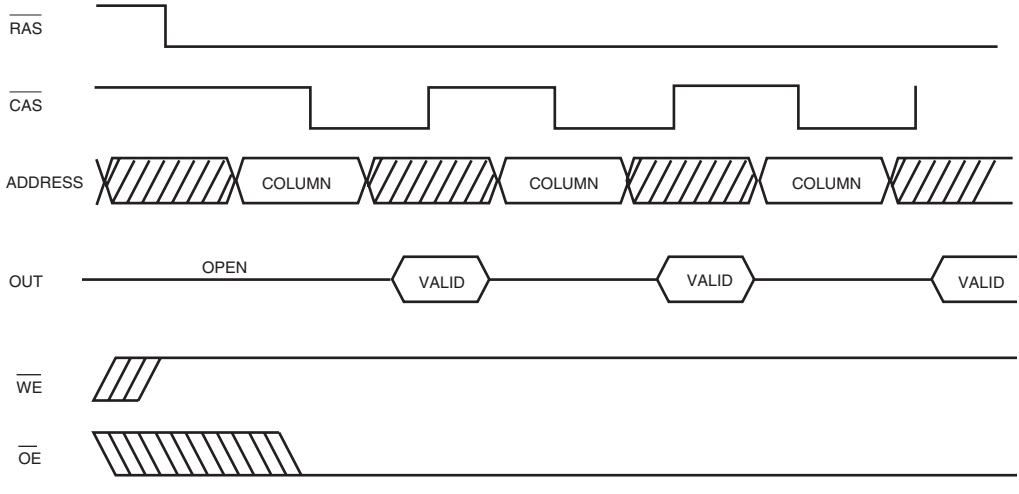


FIGURE 8.22 Page mode read cycle.

© 2006 by Taylor & Francis Group, LLC

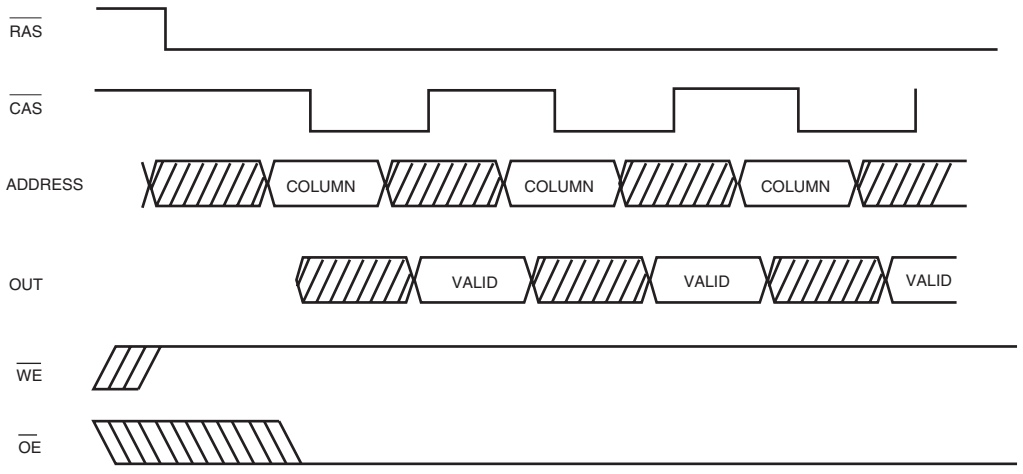


FIGURE 8.23 EDO-page mode read cycle.

Refreshing the DRAM

Row Address Strobe- (RAS-) Only Refresh

This type of refresh is done row by row. As a row is selected by providing the row address and strobing RAS, all memory cells in the row are refreshed in parallel. It will take as many cycles as the number of rows in the memory to refresh the entire device. For example, an 1M × 1 DRAM, which is built with 1024 rows and columns will take 1024 cycles to refresh the device. To reduce the number of refresh cycles, memory arrays are sometimes arranged to have less rows and more columns. The address, however, is nevertheless multiplexed as two evenly divided words (in the case of 1M × 1 DRAM the address word width is 10 b each for rows and columns). The higher order bits of address lines are used internally as column address lines and they are ignored during the refresh cycle. No CAS signal is necessary to perform the RAS-only refresh. Since the DRAM output buffer is enabled only when CAS is asserted, the data bus is not affected during the RAS-only refresh cycles.

Hidden Refresh

During a normal read cycle, RAS and CAS are strobed after the respective row and column addresses are supplied. Instead of restoring the CAS signal to high after the read, several RASs may be asserted with the corresponding refresh row address. This refresh style is called the hidden refresh cycles. Again, since the CAS is strobed and not restored, the output data are not affected by the refresh cycles. The number of refresh cycles performed is limited by the maximum time that CAS signal may be held asserted.

Column Address Strobe (CAS) Before RAS Refresh (Self-Refresh)

To simplify and speed up the refresh process, an on-chip refresh counter may be used to generate the refresh address to the array. In such a case, a separate control pin is needed to signal to the DRAM to initiate the refresh cycles. However, since in normal operation RAS is always asserted before CAS for read and write, the opposite condition can be used to signal the start of a refresh cycle. Thus, in modern self-refresh DRAMs, if the control signal CAS is asserted before the RAS, it signals the start of refresh cycles. We call this CAS-before-RAS refresh, and it is the most commonly used refresh mode in 1-Mb DRAMs. One discrepancy needs to be noted. In this refresh cycle the $\overline{\text{WE}}$ pin is a *don't care* for the 1-Mb chips. However, the 4 Mb specifies the CAS_Before_RAS refresh mode with $\overline{\text{WE}}$ pin held at high voltage. A CAS-before-RAS cycle with $\overline{\text{WE}}$ low puts the 4 Meg into the JEDEC-specified test mode (WCBR). In contrast, the 1 Meg test mode is entered by applying a high to the test pin.

© 2006 by Taylor & Francis Group, LLC

All of the mentioned three refresh cycles can be implemented on the device in two ways. One method utilizes a distributed method, the second uses a wait and burst method. Devices using the first method refresh the row at a regular rate utilizing the CBR refresh counter to turn on rows one at a time. In this type of system, when it is not being refreshed, the DRAM can be accessed, and the access can begin as soon as the self-refresh is done. The first CBR pulse should occur within the time of the external refresh rate prior to active use of the DRAM to ensure maximum data integrity and must be executed within three external refresh rate periods. Since CBR refresh is commonly implemented as the standard refresh, this ability to access the DRAM right after exciting the self-refresh is a desirable advantage over the second method. The second method is to use an internal burst refresh scheme. Instead of turning on rows at regular interval, a sensing circuit is used to detect the voltage of the storage cells to see if they need to be refreshed. The refresh is done with a series of refresh cycles one after another until all rows are completed. During the refresh other access to the DRAM is not allowed.

8.5 Error Detection and Correction

Most DRAMs require a read parity bit for two reasons. First, alpha particle strikes disturb cells by ionizing radiation, resulting in lost data. Second, when reading DRAM, the cell's storage mechanism capacitively shares its charge with the bit-line through an enable (select) transistor. This creates a small voltage differential to be sensed during read access. This small voltage difference can be influenced by other close by bit-line voltages and other noises. To have even more reliable memory, error correction code may be used. One of the error correction methods is called the Hamming code, which is capable of correcting any 1-b error.

Defining Terms

Dynamic random access memory (DRAM): This memory is dynamic because it needs to be refreshed periodically. It is random access because it can be read and written.

Memory access time: The time between a valid address supplied to a memory device and data becoming ready at output of the device.

Memory cycle time: The time between subsequent address issues to a memory device.

Memory hierarchy: Organize memory in levels to make the speed of memory comparable to the processor.

Memory read: The process of retrieving information from memory.

Memory write: The process of storing information into memory.

ROM: Acronym for read-only memory.

Static random-access memory (SRAM): This memory is static because it needs not to be refreshed. It is random access because it can be read and written.

References

Alexandridis, N. 1993. *Design of Microprocessor-Based Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Chang, S.S.L. 1980. Multiple-read single-write memory and its applications. *IEEE Trans. Comp.* C-29(8).

Chou, N.J. et. al. 1972. Effects of insulator thickness fluctuations on MNOS charge storage characteristics. *IEEE Trans. Elec. Dev.* ED-19:198.

Denning, P.J. 1968. The working set model for program behavior. *CACM* 11(5).

Flannigan, S. and Chappell, B. 1986. *J. Solid St. Cir*.

Fukuma, M. et al. 1993. Memory LSI reliability. *Proc. IEEE* 81(5). May.

Hamming, R.W. 1950. Error detecting and error correcting codes. *Bell Syst. J.* 29 (April).

Katz, R.H. et al. 1989. Disk system architectures for high performance computing. *Proc. IEEE* 77(12).

Lundstrom, K.I. and Svensson, C.M. 1972. Properties of MNOS structures. *IEEE Trans. Elec. Dev.* ED-19:826.

Masuoka, F. et al. 1984. A new flash EEPROM cell using triple poly-silicon technology. *IEEE Tech. Dig.* IEDM: 464–467.

© 2006 by Taylor & Francis Group, LLC

Micro. *Micro DRAM Databook*.
Mukherjee, S. et al. 1985. A single transistor EEPROM cell and its implementation in a 512 K CMOS EEPROM. *IEEE Tech. Dig.* IEDM:616–619.
NEC. n.d. *NEC Memory Product Databook*.
Pohm, A.V. and Agrawal, O.P. 1983. *High-Speed Memory Systems*. Reston Pub., Reston, VA.
Prince, B. and Gunnar Due-Gundersen, G. 1983. *Semiconductor Memories*. Wiley, New York.
Ross, E.C. and Wallmark, J.T. 1969. Theory of the switching behavior of MIS memory transistors. *RCA Rev.* 30:366.
Samachisa, G. et al. 1987. A 128 K flash EEPROM using double poly-silicon technology. *IEEE International Solid State Circuits Conference*, 76–77.
Sayers. et al. 1991. *Principles of Microprocessors*. CRC Press, Boca Raton, FL.
Scheibe, A. and Schulte, H. 1977. Technology of a new *n*-channel one-transistor EAROM cell called SIMOS. *IEEE Trans. Elec. Dev.* ED-24(5).
Seiichi Aritome, et al. 1993. Reliability issues of flash memory cells. *Proc. IEEE* 81(5).
Shoji, M. 1989. *CMOS Digital Circuit Technology*. Prentice-Hall, Englewood Cliffs, NJ.
Slater, M. 1989. *Design of Microprocessor-based Systems*.

Further Information

More information on basic issues concerning memory organization and memory hierarchy can be found in Pohm and Agrawal (1983). Prince and Due-Gunderson (1983) provides a good background on the different memory devices. Newer memory technology can be found in memory device databooks such as Mukherjee et al. (1985) and the NEC databook. *IEEE Journal on Solid-State Circuits* publishes an annual special issue on the Internation Solid-State Circuits Conference. This conference reports the current state-of-the-art development on most memory devices such as DRAM, SRAM, EEPROM, and flash ROM. Issues related to memory technology can be found in the *IEEE Transactions on Electron Devices*. Both journals have an annual index, which is published at the end of each year (December issue).

© 2006 by Taylor & Francis Group, LLC

9 Microprocessors

9.1 Introduction 9-1
9.2 Architecture Basics 9-1
Complex Instruction Set Computer (CISC) and Reduced Instruction Set Computer (RISC) Processors • Logical Similarity • Short Chronology of Microprocessor Development • The Intel Family of Microprocessors • The Motorola Family of Microprocessors • RISC Processor Development

James G. Cottle

9.1 Introduction

In the simplest sense, a *microprocessor* may be thought of as a central processing unit (CPU) on a chip. Technical advances of microprocessors evolve quickly and are driven by progress in ultra large-scale integrated/very large-scale integrated (ULSI/VLSI) physics and technology; fabrication advances, including reduction in feature size; and improvements in architecture.

Developers of microprocessor-based systems are interested in cost, performance, power consumption, and ease of programmability. The latter of these is, perhaps, the most important element in bringing a product to market quickly. A key item in the ease of programmability is availability of program development tools because it is these that save a great deal of time in small system implementation and make life a lot easier for the developer. It is not unusual to find small electronic systems driven by microcontrollers and microprocessors that are far more complex than need be for the project at hand simply because the ease of development on these platforms offsets considerations of cost and power consumption. Development tools are, therefore, a tangible asset to the efficient implementation of systems, utilizing microprocessors.

Often, a more general purpose microprocessor has a companion *microcontroller* within the same family. The subject of microcontrollers is a subset of the broader area of *embedded systems*. A microcontroller contains the basic architecture of the parent microprocessor with additional on-chip special purpose hardware to facilitate easy small system implementation. Examples of special purpose hardware include analog to digital (A/D) and digital to analog (D/A) converters, timers, small amounts of on-chip memory, serial and parallel interfaces, and other input output specific hardware. These devices are most cost efficient for development of the small electronic control system, since all externally needed hardware is already designed onto the chip. Component count external to the microcontroller is therefore kept to a minimum. In addition, the advanced development tools of the parent microprocessor are often available for programming. Therefore, a potential product may be brought to market much faster than if the developer used a more general purpose microprocessor, which often requires a substantial amount of external support hardware.

9.2 Architecture Basics

In the early days of mainframe computers, only a few instructions (e.g., addition, load accumulator, store to memory) were available to the programmer and the CPU had to be patched, or its configuration changed, for various applications. Invention of microprogramming, by Maurice Wilkes in 1949, unbound

© 2006 by Taylor & Francis Group, LLC

the instruction set. Microprogramming made possible more complex tasks by manipulating the resources of the CPU (its registers, arithmetic logic unit, and internal buses) in a well-defined, but programmable, way. In this type of CPU the microprogram, or *control store*, contained the realization of the semantics of native machine instructions on a particular set of CPU internal resources. Microprogramming is still in strong use today. Within the microprogram resides the routines used to implement more complex instructions and addressing modes. It is this scheme that is still widely used and a key element of the complex instruction set computer(CISC) microprocessor.

Evolution from the original large mainframe computers with hard-wired logical paths and schemes for instruction manipulation toward more flexible and complex instruction sets and addressing modes was a natural one. It was driven by advances in semiconductor fabrication technology, memory design, and device physics. Advances in semiconductor fabrication technology have made it possible to place the CPU on a monolithic integrated circuit along with additional hardware. With the advent of programmable read-only memory, the microprogram was not even bound to the whims of the original designer. It could be reprogrammed at a later date if the manipulation of resources for a particular task needed to be modified or streamlined. In the simplest view, microprocessors became more and more complex with larger instruction sets and many addressing modes. All of these advances were welcomed by compiler writers, and the instruction set complexity was a natural evolution of advances in component reliability and density.

Complex Instruction Set Computer (CISC) and Reduced Instruction Set Computer (RISC) Processors

The strength of the microprocessor is in its ability to perform simple logical tasks at a high rate of speed. In *Complex instruction set computer* (CISC) microprocessors, small register sets, memory to memory operations, large instruction sets (with variable instruction lengths), and the use of microcode are typical. The basic simplified philosophy of the CISC microprocessor is that added hardware can result in an overall increase in speed. The penultimate CISC processor would have each high-level language statement mapped to a single native CPU instruction. Microcode simplifies the complexity somewhat but necessitates the use of multiple machine cycles to execute a single CISC instruction. After the instruction is decoded on a CISC machine, the actual implementation may require 10 or 12 machine cycles depending on the instruction and addressing mode used. The original trend in microprocessor development was toward increased complexity of the instruction set. Although there may be hundreds of native machine instructions, only a handful are actually used. Ironically, the CISC instruction set complexity evolves at a sacrifice in speed because its harder to increase the clock speed of a complex chip. Recently, recognition of this along with demands for increased clock speeds have yielded favor toward the reduced instruction set (RISC) microprocessor.

This trend reversal followed studies in the early 1970s that showed that although the CISC machines had plenty of instructions, only relatively few of these were actually being used by programmers. In fact, 85% of all programming consists of simple assignment instructions) (i.e., A=B).

RISC machines have very few instructions and few machine cycles to implement them. What they do have is a lot of registers and a lot of parallelism. The ideal RISC machine attempts to accomplish a complete instruction in a single machine cycle. If this were the case, a 100-MHz microprocessor would execute 100 million instructions per second. There are typically many registers for moving data to accomplish the goal of reduced machine cycles. There is, therefore, a high degree of parallelism in a RISC processor. On the other hand, CISC machines have relatively few registers, depending on multiple machine cycle manipulation of data by the microprogram. In a CISC machine, the microprogram handles a lot of complexity in interpreting the native machine instruction. It is not uncommon for an advanced microprocessor of this type to have over 100 native machine instructions and a slew of addressing modes.

The two basic philosophies, CISC and RISC, are ideal concepts. Both philosophies have their merits. In practice, microprocessors typically incorporate both philosophical schemes to enhance performance and speed. A general summary of RISC vs CISC is as follows. CISC machines depend on complexity of programming in the microprogram, thereby simplifying the compiler. Complexity in the RISC machine is realized by the compiler itself.

© 2006 by Taylor & Francis Group, LLC

Logical Similarity

Microprocessors may often be highly specialized for a particular application. The external logical appearance of all microprocessors, however, is the same. Although the package itself will differ, it contains connection pins for the various control signals coming from or going to the microprocessor. These are connected to the system power supply, external logic, and hardware to made up a complete computing system. External memory, math-processors, clock generators, and interrupt controllers are examples of such hardware and all must have an interface to the generic microprocessor. A diagram of a typical microprocessor pinout and the common grouping of these control signals is shown in Fig. 9.1 and microprocessor printout for Intel® 8086 and 8088 processors are shown Fig. 9.2. Externally, there are signals, which may be grouped as addressing, data, bus arbitration and control, coprocessor signals, interrupts, status, and miscellaneous connections. In some cases, multiple pins are used for connections such as the case for the address and data buses.

Just as the logical connections of the microprocessor are the same from family to family, so are the basic instructions. For instance, all microprocessors have an add instruction, a subtract instruction, and a memory read instruction. Although logically, these instructions all are the same, the particular microprocessor realizes the semantics of the instruction (e.g., ADD) in a very unique way. This implementation of the ADD depends on the internal resources of the CPU such as how many registers are available, how many internal buses there are, and whether the data and address information may be separated on travel along the same internal paths.

To understand how a microprogram realizes the semantics of a native machine language instruction in a CISC machine, refer to Fig. 9.3. The figure illustrates a very simple CISC machine with a single bus, which is used to transfer both memory and data information on the inside of the microprocessor. Only one instruction may be operated on at a time with this scheme, and it is far simpler than most microprocessors available today, but it illustrates the process of instruction implementation. The process begins by a procedure (a routine in control store called the instruction fetch) that fetches the next contiguous instruction in memory and places its contents in the instruction register for interpretation. The program counterregister contains the address of this instruction, and its contents are placed in the memory address

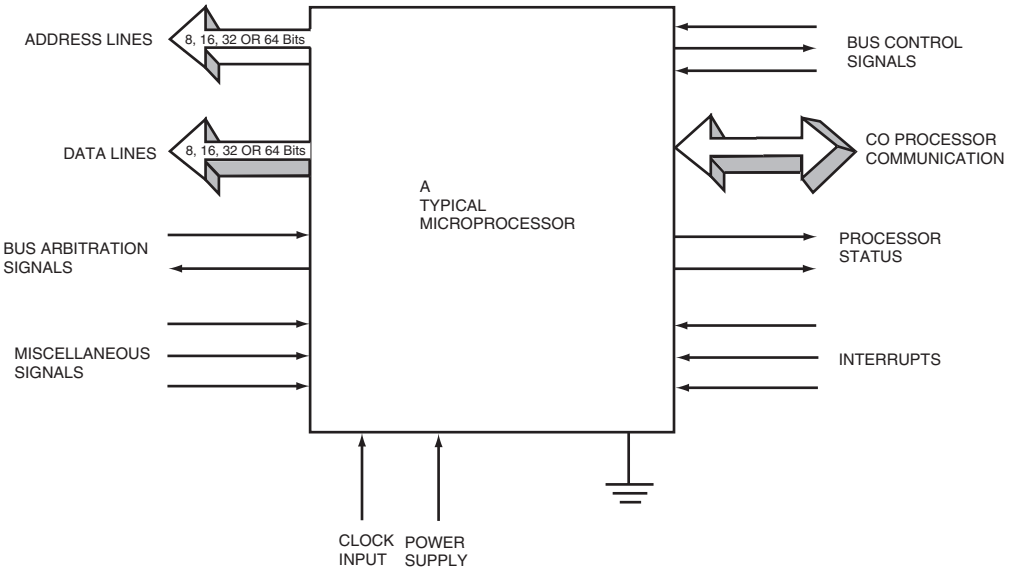


FIGURE 9.1 The pins connecting all microprocessors to external hardware are logically the same. They contain pins for addressing, data bus arbitration and control, coprocessor signals and interrupts, and those pins needed to connect to the power supply and clock generating chips.

© 2006 by Taylor & Francis Group, LLC

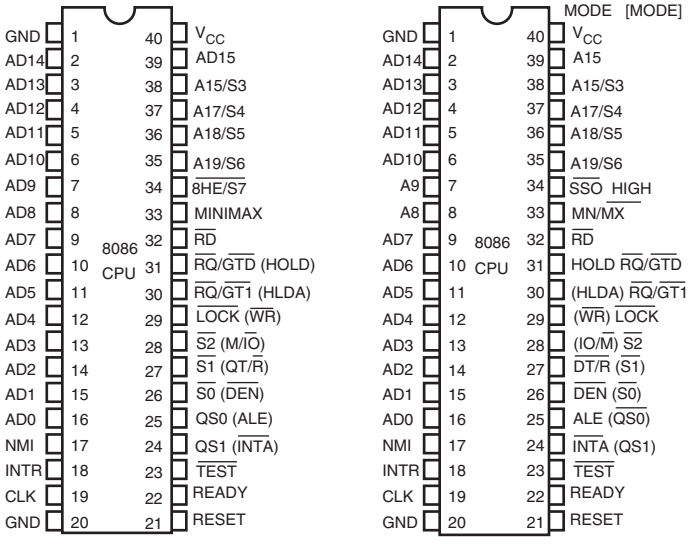


FIGURE 9.2 Microprocessor chip pinouts for the Intel 8086 (left) and the 8088 (right).

register so that execution of a read command to the memory will cause, some time later, the contents of the next instruction to appear in the memory data register. These contents are then moved along the bus to the instruction register to be decoded. Meanwhile, the program counter is appropriately incremented and contains the address of the next instruction to be fetched. The contents of the instruction register for the current instruction contain the opcode and certain address information regarding the operands associated with the coded operation. For example, if the fetched instruction were an add, the instruction register would contain the coded bits indicating an add, and the information relevant to obtaining the operands for the addition. The number of bits for the machine is constant (such as 8-,16-, or 32-b instructions) but the number of bits dedicated to the opcode and operand fields may vary to accommodate instructions with single or multiple operands, different addressing modes, and so forth. The opcode is almost always a coded address referring to a position in the microprogram of the microprocessor. For the case of the add instruction, the opcode indicates a place in microprogram memory that contains the logic code needed to implement the add. Several machine cycles are necessary for the add instruction, including those steps needed to fetch the instruction itself. The number of these steps depends on the particular microprocessor. That is, all machines have an add instruction. The microprogram contains the realization of the add on the specific microprocessor's architecture with its individual set of hardware resources.

Specific steps (micro-operations), which direct the microprocessor to fetch the next instruction from memory, are included with every native instruction. The instruction fetch procedure is an integral part of every native instruction and represents the minimum number of machine cycles that a microprocessor must go through to implement even the simplest instruction. Even a halt instruction, therefore, requires several machine cycles to implement.

The advantages of the CISC scheme are clear. With a large number of instructions available, the programmer of a compiler or high-level language interpreter has a relatively easy task and a lot of tools at hand. These include addressing mode schemes to facilitate relative addressing, indirect addressing, and modes such as autoincrementing or decrementing. In addition, there are typically instructions for memory to memory data transfers. Ease of programming, however, does not come without a price. The disadvantages of the CISC scheme are, therefore, relatively clear. Implementation of a particular instruction is bound to the microprogram, may take too many machine cycles, and may be relatively slow. The CISC architectural scheme also requires a lot of real estate on the semiconductor chip. Therefore, size and speed (which are really one and the same) are sacrificed for programming simplicity.

© 2006 by Taylor & Francis Group, LLC

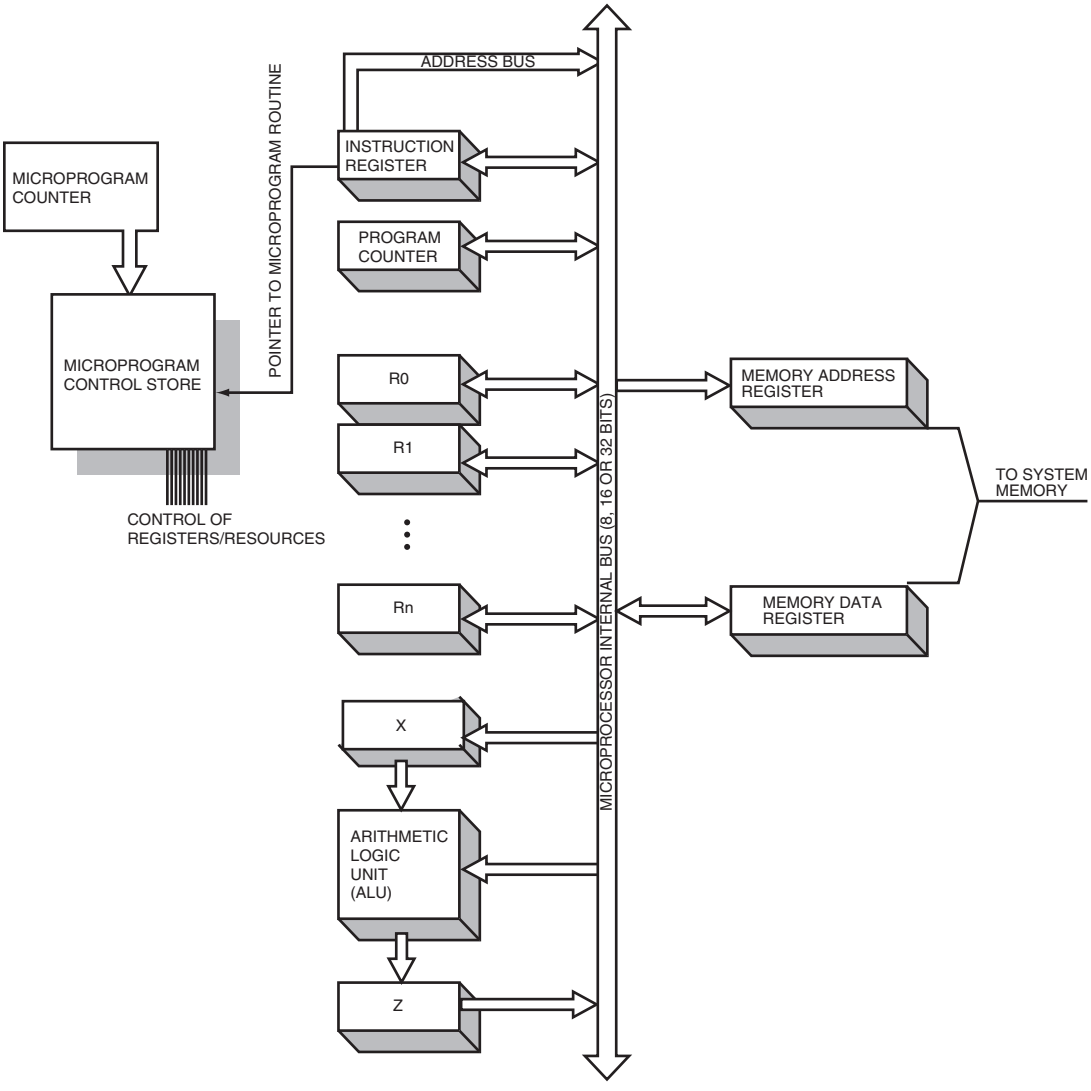


FIGURE 9.3 The basic single bus computer. This CISC machine is characterized by multiple registers linked by a bus, which carries both address and data information to and from the various registers (see text). Control lines gating the input and output of each register are governed by the information contained in the control store.

The von Neumann architecture was once the basis of most all CISC machines. This scheme is characterized by a common bus for data and address flow linking a small number of registers. The number of registers in the von Neumann machine varies depending on design, but typically consists of about 10 registers including those for general program data and addresses as well as special purposes such as instruction storage, memory addressing, and latching registers for the arithmetic logic unit (ALU). The basic architecture of the von-Neumann machine is illustrated in Figure 8.68. Recent advances, however, have led to a departure of CISC machines from the basic single bus system.

Short Chronology of Microprocessor Development

The first single chip CPU was the Intel 4004 developed for calculators. It processed data in 4 bits, and its instructions were 8 bits long. Program and data were separate. The 4004 had 46 instructions, a 4 level stack, 12-b program counter, and 16, 4-b registers. Later in 1972, the 4004's successor, the 8008, was introduced,

© 2006 by Taylor & Francis Group, LLC

which was followed by the 8080 in 1974. The 8080 had a 16-b address bus and an 8-b data bus, seven, 8-b registers, a 16-b stack pointer to memory, and a 16-b program counter. It also had 256 input/output (I/O) ports so that I/O devices did not take up memory space and could be addressed more directly. The design was updated in 1976 (the 8085) to only require a +5 V supply.

Zilog in July 1976 introduced the Z-80, which was intended to be an improved 8080. It also used 8-b data and 16-b address, could execute all of the opcodes of the 8080 and added 80 more instructions. The register set was doubled and consisted of two banks, which could be switched. Two index registers (IX and IY) allowed for more complex memory instructions. Probably the most successful feature of the Z-80 was its memory interface. Until its introduction, dynamic random-access memory (RAM) had to be refreshed with rather complex external circuitry, which made small computing systems more complex and expensive. The Z-80 was the first chip to incorporate this refreshing capability on-chip, which increased its popularity among system developers. The Z-8 was an embedded processor similar to the Z-80 with on-chip RAM and read-only memory (ROM). It was available in clock speeds to 20 MHz and was used in a variety of small microprocessor-based control systems.

The next processor of note in the chronology was the 6800 in 1975. Although introduced by Motorola, MOS Technologies gained popularity through introducing its 650x series, principally, the 6502, which was used in early desktop computers (Commodores, Apples, and Ataris). The 6502 had very few registers and was principally an 8-b processor with a 16-b address bus. The Apple II, one of the first computers introduced to the mainstream consumer market, incorporated the 6502. Subsequent improvements in the Apple line of micros were downward compatible with the 6502 processor. The extension to the 6502 came in 1977 when Motorola introduced the 6809 with two 8-b accumulators, which could combine mathematics operations in a single 16-b combination. It had 59 instructions. Members of the 6800 family live on in embedded microcontrollers such as the 68HC05 and 68HC11. These microcontrollers are still popular for small control systems. The 68HC11 was extended to 16-b and named the 68HC16. Radiation hardened versions of the 68HC11 have been used in communications satellites.

Advanced micro devices (AMD) introduced a 4-b bit-sliced microprocessor, the Am2901. Bit-sliced processors were modular in that they could be assembled to form larger word sizes. The Am2901 had a 4-b ALU; 16, 4-b registers; and the hardware to connect carry/borrow signals between adjacent modules. In 1979, AMD developed the first floating point coprocessor for microprocessors. The AMD 9511 arithmetic circuit was used in some CP/M, Z-80-based systems and some systems based on the S-100 bus.

Around 1976, competition was heating up for the 16-b microprocessor market. The Texas Instruments (TI) TM9900 was one of the first truly 16-b microprocessors and was designed as a single chip version of the TI 990 minicomputer. The TM9900 had two 16-b registers, good interrupt handling capability, and a decent instruction set for compiler developers. An embedded version (the TMS 9940) was also produced by TI. In 1976, the stage was being set for IBM's choice of a microprocessor for its IBM-PC line of personal computers. Several 16-b microprocessors around at the time had much more powerful features and more straightforward open memory architectures (such as Motorola's 68000). It is rumoured that IBM's own engineers wanted to use the 68000 but at the time IBM had already negotiated the rights to the Intel 8086. Apparently, the choice of the 8-b 8088 was a cost decision because the 8088 could have used the lower cost support chips associated with the 8085, whereas 68000 components were more expensive and not readily available.

Around 1976, Zilog introduced the Z-8000 (shortly after the 8086 by Intel). It was a 16-b microprocessor that had the capability of addressing up to 23-b of address data. The Z-8000 had 16, 16-b registers. The first 8 could be used as 16, 8-b registers, or all 16 could be used as 8, 32-b registers. This offered great flexibility in programming and for arithmetic calculations. Its instruction set included a 32-b multiply and divide instruction. It, also, like the Z-80 had memory refresh circuitry built into the chip. Probably most important, however, in the CPU chronology, is that the Z-8000 was the first microprocessor to incorporate two different modes of operation. One mode was strictly reserved for use by the operating system. The other mode was a general purpose user mode. The use of this scheme improved stability, in that the user could not crash the system as easily, and opened up the possibility of porting the chip towards multitasking, multiuser operating systems such as UNIX.

© 2006 by Taylor & Francis Group, LLC

The Intel Family of Microprocessors

Intel was the first to develop a CPU on a chip in 1971 with the 4004 microprocessor. This chip, along with the 8008, was commissioned for calculators and terminal control. Intel did not expect the demand for these units to be high and several years later, developed a more general purpose microprocessor, the 8080, and a similar chip with more onboard hardware, the 8085. These were the industry's first truly general CPU's available to be integrated into microcomputing systems. The first 16-b chip, the 8086, was developed by Intel in 1978 and was the first industry entry into the realm of the 16-b processors. A companion mathematic coprocessor, the 8087, was developed for calculations requiring higher precision than the 16-b registers the 8086 would offer. Shortly after developing the 8086 and the 16-b address/8-b data version the 8088, IBM chose the 8088 for its IBM PC microcomputers. This decision was a tremendous boon to Intel and its microprocessor efforts. In some ways, it has also made the 80×86 family a victim of its early success, as all subsequent improvements and moves to larger data and address bus CPUs have had to contend with downward compatibility.

The 80186 and 80188 microprocessors were, in general, improvements to the 8086 and 8088 and incorporated more on-chip hardware for input and output support. They were never widely used, however, most likely due to the masking effect of the 8088's success in the IBM PC. The 80186 is architecturally identical to the 8086 but also contains a clock generator, a programmable controlled interrupt, three 16-b programmable timers, two programmable DMA controllers, a chip select unit, programmable control registers, a bus interface unit, and a 6-byte prefetch queue. The 80188 scheme is the same, with the exception that it only has an 8-b external data bus and a 4-byte prefetch queue.

None of the processors of the Intel family up to the beginning of the 1980s had the ability to address more than 1 megabyte of memory. The 80286, a 68-pin microprocessor, was developed to cater to the needs of systems and programs that were evolving, in a large part, due to the success of the 8088. The 80286 increased the available address space of the Intel microprocessor family to 16 megabytes of memory. Also, beginning with the 80286, the data and address lines external to the chip were not shared. In earlier chips, the address pins were multiplexed with the data lines. The internal architecture to accomplish this was a bit cumbersome, however, was kept so as to include downward compatibility with the earlier CPUs. Despite the scheme's unwieldiness, the 80286 was a huge success.

In a decade, the evolution of the microprocessor had advanced from its earliest beginnings (with a 4-b CPU) to a true 16-b microprocessor. Many everday computing tasks were off loaded from mainframes to desktop machines. In 1985 Intel developed a true 32-b processor on a chip, the 80386. It was downward compatible with all object codes back to the 8008 and continued to lock Intel into the rather awkward memory model developed in the 80286. At Motorola, the 68000 in some ways had a far more simple and straightforward open address space and was a serious contender for heavy computing applications being ported to desktop machines from their larger, mainframe cousins. It is for this reason that even today the 68000 is found systems requiring compatibility with UNIX operating systems. The 80386 was, nevertheless, highly successful. The 80386SX was a version of the 80386 developed with an identical package to the 80286 and meant as an upgrade for existing 80286 systems. It did so by upgrading the address but to 32 b but maintained the 16-b data bus of the 80286. A companion mathematic coprocessor (a floating point math unit (FPU)), the 80387, was developed for use with the 80386.

The 80386's success prompted other semiconductor companies (notably AMD and Cyrix) to piggyback on its success by offering clones of the processors and thus alternative sources for its end users and system developers. With Intel's addition of its 80486 in 1989, including full **pipelines**, on-chip **caching** and an integrated rather than separate floating point processor, competition for the chips, popularity was fierce. In late 1993, Intel could no longer protect the next subsequent name in the series (the 80586). It trademarked the Pentium name to its 80586 processor. Because of its popularity, the 80×86 line is the most widely cloned.

The Motorola Family of Microprocessors

Alongside of Intel's development of the 8080, Motorola developed the 6800, 8-b microprocessor. In the early 1970s, the 6800 was used in many embedded industrial control systems. It was not until 1979 however,

© 2006 by Taylor & Francis Group, LLC

that Motorola introduced its 16-b entry into the industry, the 68000. The 68000 was designed to be far more advanced than the 8086 microprocessor by Intel in several ways. All internal registers were 32-b wide, and it had the benefit of being able to address all 16 megabytes of external memory without the segmentation schemes utilized in the Intel series. This nonsegmented approach meant that the 68000 had no segment registers, and each of its instructions could address the complete complement of external memory.

The 68000 was the first 16-b microprocessor to incorporate 32-b internal registers. This asset allowed its selection by designers who set out to port sophisticated operating systems to desktop computers. In some ways, the 68000 was ahead of its time. If IBM had chosen the 68000 series as the core chip for its personal computers, the present state-of-the-art of the desktop machine would be radically different. The 68000 was chosen by Apple for its MacIntosh computers. Other computer manufacturers, including Amiga and Atari chose it for its flexibility and its large internal registers.

In 1982, Motorola marketed another chip, the 68008, which was a stripped down version of the 68000 for low-end, low-cost products. The 68008 only had the capability to address 4 megabytes of memory, and its data bus was only 8-b wide. It was never very popular and certainly did not compete well with the 8088 by Intel (which was chosen for the IBM PC computers).

Advanced operating systems were ideal for the 68000 except that the chip had no capability for supporting virtual memory. For this reason, Motorola developed the 68010, which had the capability to continue an instruction after it had been suspended by a bus error. The 68012 was identical to the 68000 except that it had the capability to address 2 gigabytes of memory with its 30 address bus pins.

One of the most successful microprocessors introduced by Motorola was the 68020. It was introduced in 1984 and was the industry's first true 32-b microprocessor. Along with the 32-b registers standard to the 68000 series, it has the capability of addressing 4 gigabytes of memory and a true 32-b wide data bus. It is still widely used.

The 68020 contains an internal 256 byte cache memory. This is an instruction cache, holding up to 64 instructions of the long-word type. Direct access to this cache is not allowed. It serves only as an advance *prefetch queue* to enable the 68020 to execute tight loops of instructions with any further instruction fetches. Since an instruction fetch takes time to process, the presence of the 256-byte instruction cache in the 68020 is a significant speed enhancement.

The cache treatment was expanded in the 68030 to include a 256-byte data cache. In addition, the 68030 includes an onboard paged memory management unit (PMMU) to control access to virtual memory. This is the primary difference in the 68030 and the 68020. The PMMU is available as an extra chip (the 68851) for the 68020 but included on the same chip with the 68030. The 68030 also includes an improved bus interface scheme. Externally, the connections of the 68020 and the 68030 are very nearly the same. The 68030 is available in two speeds, the MC68030RC16 at 16 MHz and the MC68030RC20 with a 20-MHz clock.

The 68000 featured a supervisor and user mode. It was designed for expansion and could fetch the next instruction during an instruction's execution. (This represents 2-stage pipelining.) The 68040 had 6-stages of pipelining. The advances in the 680×0 series continued toward the 68060 in late 1994, which was a ***superscalar*** microprocessor similar to the Intel Pentium. It truly represents a merging of the two CISC and RISC philosophies of architecture. The 68060 10-stage pipeline translates 680×0 instructions into a decorded RISC-like form and uses a resource renaming scheme to reorder the execution of the instructions. The 68060 includes power saving features that can be shutdown and operates off of a 3.3-V power supply (again similar to the Intel Pentium processors).

RISC Processor Development

The major development efforts for RISC processors were led by the University of California at Berkeley and the Stanford University designs. Sun Microsystems developed the Berkeley version of the RISC processor [scalable processor architecture (SPARC)] for their high-speed workstations. This, however, was not the first RISC processor. It was preceeded by the MIPS R2000 (based on the Stanford University design), the Hewlett Packard PA-RISC CPU, and the AMD 29000.

© 2006 by Taylor & Francis Group, LLC

The AMD 29000 is a RISC design, which follows the lead of the Berkeley scheme. It has a large set of registers spilt into local and global sets. The 64 global registers reduced instruction set processors were developed following the recognition that many of the CISC complex instructions were not being used.

Defining Terms

Cache: Small amount of *fast* memory, physically close to CPU, used as storage of a block of data needed immediately by the processor. Caches exist in a memory hierarchy. There is a small but very fast L1 (level one) cache; if that misses, then the access is passed on to the bigger but slower L2 (level two) cache, and if that misses, the access goes to the main memory (or L3 cache, if it exists).

Pipelining: A microarchitecture technique that divides the execution of an instruction into sequential steps. Pipelined CPUs have multiple instructions executing at the same time but at different stages in the machine. Or, the act of sending out an address before the data is actually needed.

Superscalar: Capable of executing multiple instructions in a given clock cycle. For example, the Pentium processor has two execution pipes (U and V) so it is superscalar level 2. The Pentium Pro processor can dispatch and retire three instructions per clock so it is superscalar level 3.

References

The Alpha 21164A: Continued performance leadership. 1995. *Microprocessor Forum*.

Internal architecture of the Alpha 21164 microprocessor. 1995. *CompCon 95*.

A 300 MHz quad-issue CMOS RISC microprocessor (21164). 1995. In *ISSC 95*, pp. 182–183.

A 200 MHz 64 b dual-issue CMOS microprocessor (21064). 1992. In *ISSC 92*, pp. 106–107.

Hobbit: A high performance, low-power microprocessor. 1993. *CompCon 93*, pp. 88–95.

MIPS R10000 superscalar microprocessor. 1995. *Hot Chips VII*.

The impact of dynamic execution techniques on the data path design of the P6 processor. 1995. *Hot Chips VII*.

A 0.6 μm BiCMOS processor with dynamic execution (P6). 1995. In *ISSC 95*, pp. 176–177.

A 3.3 v 0.6 μm BiCMOS superscalar processor (Pentium). 1994. In *ISSC 94*, pp. 202–203.

An overview of the Intel Pentium processor. 1993. In *CompCon 93*, pp. 60–62.

Superscalar architecture of the P5-$\times$86 next generation processor. 1992. *Hot Chips IV*.

A 93 MHz x86 microprocessor with on-chip L2 cache controller (N586). 1995. In *ISSC 95*, pp. 172–173.

The AMD K5 processor. 1995. *Hot Chips VII*.

The PowerPC620 microprocessor: A high performance superscalar RISC microprocessor. *CompCon 95*.

A new powerPC microprocessor for the low power computing marker (602). 1995. *CompCon 95*.

133 MHz 64 b four-issue CMOS microprocessor (620). In *ISSC 95*. 1995. pp. 174–175.

The powerPC 604 RISC microprocessor. 1994. *IEEE Micro*. (Oct.).

The powerPC user instruction set architecture. 1994. *IEEE Micro*. (Oct.).

PowerPC 604. 1994. *Hot Chips VI*.

The powerPC 603 microprocessor: A low power design for portable applications. 1994. In *CompCon 94*, pp. 307–315.

A 3.0 W 75SPECint92 85SPECfp92 superscalar RISC microprocessor (603). 1994. In *ISSC 94*, pp. 212–214.

601 powerPC microprocessor. 1993. *Hot Chips V*.

The powerPC 601 microprocessor. 1993. In *Compcon 93*, pp. 109–116.

The great dark cloud falls: IBM's choice. In *Great Microprocessors of the Past and Present* (on-line) Sec. 3. http:// www.cpu info.berkeley. edu.

Further Information

In the fast changing world of the microprocessor, obsolecence is a fact of life. In fact, the marketing data shows that, with each introduction of a new generation, the time spent ramping up a new technology and ramping down the old gets shorter. There is more need for accuracy in design and development to

© 2006 by Taylor & Francis Group, LLC

avoid errors such as Intel experienced with their Pentium Processor with their floating point mathematic computations in 1994. For the small system developer or user of microprocessors, there is an important need to keep abreast of newer, more enhanced chips with better capabilities. Luckily, there is an excellent way to keep up to date.

The CPU Information Center at the University of California, Berkeley maintains an excellent, up to date compilation of microprocessors and microcontrollers, their architectures, and specifications on the World Wide Web (WWW). The site includes chip size and pinout information, tabular comparisons of microprocessor performance and architecture (such as that shown in the tables in this chapter), and references. It is updated regularly and serves as an excellent source for the small systems developer. Some of the information contained in this chapter was obtained and used with permission from that Web site. See http://www. infopad. berkeley.edu for an excellent, up to date summary.

© 2006 by Taylor & Francis Group, LLC

10

D/A and A/D Converters

10.1 Introduction **10**-1
10.2 D/A and A/D Circuits **10**-1
D/A and A/D Converter Performance Criteria • D/A Conversion Processes • D/A Converter ICs • A/D Conversion Processes • A/D Converter ICs • Grounding and Bypassing on D/A and A/D ICs • Selection Criteria for D/A and A/D Converter ICs

Susan A. Garrod

10.1 Introduction

Digital-to-analog (D/A) conversion is the process of converting digital codes into a continuous range of analog signals. *Analog-to-digital (A/D) conversion* is the complementary process of converting a continuous range of analog signals into digital codes. Such conversion processes are necessary to interface real-world systems, which typically monitor continuously varying analog signals, with digital systems that process, store, interpret, and manipulate the analog values.

D/A and A/D applications have evolved from predominately military-driven applications to consumer-oriented applications. Up to the mid-1980s, the military applications determined the design of many D/A and A/D devices. The military applications required very high performance coupled with hermetic packaging, radiation hardening, shock and vibration testing, and military specification and record keeping. Cost was of little concern, and "low power" applications required approximately 2.8 W. The major applications up the mid-1980s included military radar warning and guidance systems, digital oscilloscopes, medical imaging, infrared systems, and professional video.

The applications requiring D/A and A/D circuits today have different performance criteria from those of earlier years. In particular, low power and high speed applications are driving the development of D/A and A/D circuits, as the devices are used extensively in battery-operated consumer products.

10.2 D/A and A/D Circuits

D/A and A/D conversion circuits are available as integrated circuits (ICs) from many manufacturers. A huge array of ICs exists, consisting of not only the D/A or A/D conversion circuits, but also closely related circuits such as sample-and-hold amplifiers, analog multiplexers, voltage-to-frequency and frequency-to-voltage converters, voltage references, calibrators, operation amplifiers, isolation amplifiers, instrumentation amplifiers, active filters, DC-to-DC converters, analog interfaces to digital signal processing systems, and data acquisition subsystems. Data books from the IC manufacturers contain an enormous amount of information about these devices and their applications to assist the design engineer.

© 2006 by Taylor & Francis Group, LLC

The ICs discussed in this chapter will be strictly the D/A and A/D conversion circuits. The ICs usually perform either D/A or A/D conversion. There are serial interface ICs, however, typically for digital signal processing applications, that perform both A/D and D/A processes.

D/A and A/D Converter Performance Criteria

The major factors that determine the quality of performance of D/A and A/D converters are *resolution, sampling rate, speed,* and *linearity*.

The *resolution* of a D/A circuit is the smallest change in the output analog signal. In an A/D system, the resolution is the smallest change in voltage that can be detected by the system and can produce a change in the digital code. The resolution determines the total number of digital codes, or *quantization levels*, that is recognized or produced by the circuit.

The *resolution* of a D/A or A/D IC is usually specified in terms of the bits in the digital code or in terms of the least significant bit (LSB) of the system. An n-bit code allows for 2^n quantization levels, or $2^n - 1$ steps between quantization levels. As the number of bits increases, the step size between quantization levels decreases, therefore increasing the accuracy of the system when a conversion is made between an analog and digital signal. The system resolution can be specified also as the voltage step size between quantization levels. For A/D circuits, the resolution is the smallest input voltage that is detected by the system.

The *speed* of a D/A or A/D converter is determined by the time it takes to perform the conversion process. For D/A converters, the speed is specified as the *settling time*. For A/D converters, the speed is specified as the *conversion time*. The settling time for D/A converters varies with supply voltage and transition in the digital code; thus, it is specified in the data sheet with the appropriate conditions stated.

A/D converters have a maximum *sampling rate* that limits the speed at which they can perform continuous conversions. The sampling rate is the number of times per second that the analog signal can be sampled and converted into a digital code. For proper A/D conversion, the minimum sampling rate must be at least two times the highest frequency of the analog signal being sampled to satisfy the Nyquist sampling criterion. The conversion speed and other timing factors must be taken into consideration to determine the maximum sampling rate of an A/D converter. **Nyquist A/D converters** use a sampling rate that is slightly more than twice the highest frequency in the analog signal. **Oversampling A/D converters** use sampling rates of N times rate, where N typically ranges from 2 to 64.

Both D/A and A/D converters require a voltage reference in order to achieve absolute conversion accuracy. Some conversion ICs have internal voltage references, whereas others accept external voltage references. For high-performance systems, an external precision reference is needed to ensure long-term stability, load regulation, and control over temperature fluctuations. External precision voltage reference ICs can be found in manufacturer's data books.

Measurement accuracy is specified by the converter's *linearity*. *Integral linearity* is a measure of linearity over the entire conversion range. It is often defined as the deviation from a straight line drawn between the endpoints and through zero (or the offset value) of the conversion range. Integral linearity is also referred to as *relative accuracy*. The *offset* value is the reference level required to establish the zero or midpoint of the conversion range. *Differential linearity* is the linearity between code transitions. Differential linearity is a measure of the *monotonicity* of the converter. A converter is said to be monotonic if increasing input values result in increasing output values.

The accuracy and linearity values of a converter are specified in the data sheet in units of the LSB of the code. The linearity can vary with temperature, and so the values are often specified at $+25^{\circ}$C as well as over the entire temperature range of the device.

D/A Conversion Processes

Digital codes are typically converted to analog voltages by assigning a voltage weight to each bit in the digital code and then summing the voltage weights of the entire code. A general D/A converter consists of a network of precision resistors, input switches, and level shifters to activate the switches to convert a digital

© 2006 by Taylor & Francis Group, LLC

code to an analog current or voltage. D/A ICs that produce an analog current output usually have a faster settling time and better linearity than those that produce a voltage output. When the output current is available, the designer can convert this to a voltage through the selection of an appropriate output amplifier to achieve the necessary response speed for the given application.

D/A converters commonly have a fixed or variable reference level. The reference level determines the switching threshold of the precision switches that form a controlled impedance network, which in turn controls the value of the output signal. **Fixed reference D/A converters** produce an output signal that is proportional to the digital input. **Multiplying D/A** converters produce an output signal that is proportional to the product of a varying reference level times a digital code.

D/A converters can produce bipolar, positive, or negative polarity signals. A four-quadrant multiplying D/A converter allows both the reference signal and the value of the binary code to have a positive or negative polarity. The four-quadrant multiplying D/A converter produces bipolar output signals.

D/A Converter ICs

Most D/A converters are designed for general-purpose control applications. Some D/A converters, however, are designed for special applications, such as video or graphic outputs, high-definition video displays, ultra high-speed signal processing, digital video tape recording, digital attenuators, or high-speed function generators.

D/A converter ICs often include special features that enable them to be interfaced easily to microprocessors or other systems. Microprocessor control inputs, input latches, buffers, input registers, and compatibility to standard logic families are features that are readily available in D/A ICs. In addition, the ICs usually have laser-trimmed precision resistors to eliminate the need for user trimming to achieve full-scale performance.

A/D Conversion Processes

Analog signals can be converted to digital codes by many methods, including integration, **successive approximation**, parallel (flash) conversion, **delta modulation, pulse code modulation**, and **sigma–delta conversion**. Two common A/D conversion processes are successive approximation A/D conversion and parallel or **flash A/D** conversion. Very high-resolution digital audio or video systems require specialized A/D techniques that often incorporate one of these general techniques as well as specialized A/D conversion processes. Examples of specialized A/D conversion techniques are pulse code modulation (PCM), and sigma–delta conversion. PCM is a common voice encoding scheme used not only by the audio industry in digital audio recordings but also by the telecommunications industry for voice encoding and multiplexing. Sigma–delta conversion is an oversampling A/D conversion where signals are sampled at very high frequencies. It has very high resolution and low distortion.

Successive approximation A/D conversion is a technique that is commonly used in medium- to high-speed data acquisition applications. It is one of the fastest A/D conversion techniques that requires a minimum amount of circuitry. The conversion times for successive approximation A/D conversion typically range from 10 to 300 μs for 8-b systems.

The successive approximation A/D converter can approximate the analog signal to form an n-bit digital code in n steps. The successive approximation register (SAR) individually compares an analog input voltage to the midpoint of one of n ranges to determine the value of 1 b. This process is repeated a total of n times, using n ranges, to determine the n bits in the code. The comparison is accomplished as follows. The SAR determines if the analog input is above or below the midpoint and sets the bit of the digital code accordingly. The SAR assigns the bits beginning with the most significant bit. The bit is set to a 1 if the analog input is greater than the midpoint voltage, or it is set to a 0 if it is less than the midpoint voltage. The SAR then moves to the next bit and sets it to a 1 or a 0 based on the results of comparing the analog input with the midpoint of the next allowed range. Because the SAR must perform one approximation for each bit in the digital code, an n-bit code requires n approximations.

© 2006 by Taylor & Francis Group, LLC

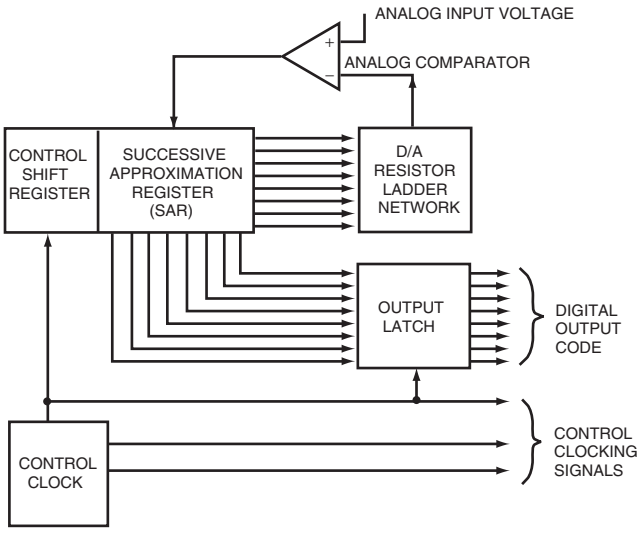


FIGURE 10.1 Successive approximation A/D converter block diagram. (*Source:* Garrod, S. and Borns, R. 1991. *Digital Logic: Analysis, Application, and Design*, p. 919. Copyright ©1991 by Saunders College Publishing, Philadelphia, PA. Reprinted by permission of the publisher.)

A successive approximation A/D converter consists of four functional blocks, as shown in Fig. 10.1: the SAR, the analog comparator, a D/A converter, and a clock.

Parallel or flash A/D conversion is used in high-speed applications such as video signal processing, medical imaging, and radar detection systems. A flash A/D converter simultaneously compares the input analog voltage to $2^n - 1$ threshold voltages to produce an n-bit digital code representing the analog voltage. Typical flash A/D converters with 8-b resolution operate at 20–100 MHz.

The functional blocks of a flash A/D converter are shown in Fig. 10.2. The circuitry consists of a precision resistor ladder network, $2^n - 1$ analog comparators, and a digital priority encoder. The resistor network establishes threshold voltages for each allowed quantization level. The analog comparators indicate whether or not the input analog voltage is above or below the threshold at each level. The output of the analog comparators is input to the digital priority encoder. The priority encoder produces the final digital output code that is stored in an output latch.

An 8-b flash A/D converter requires 255 comparators. The cost of high-resolution A/D comparators escalates as the circuit complexity increases and as the number of analog converters rises by $2^n - 1$. As a low-cost alternative, some manufacturers produce modified flash A/D converters that perform the A/D conversion in two steps to reduce the amount of circuitry required. These modified flash A/D converters are also referred to as *half-flash* A/D converters, since they perform only half of the conversion simultaneously.

A/D Converter ICs

A/D converter ICs can be classified as general-purpose, high-speed, flash, and sampling A/D converters. The *general-purpose A/D converters* are typically low speed and low cost, with conversion times ranging from 2 μs to 33 ms. A/D conversion techniques used by these devices typically include successive approximation, tracking, and integrating. The general-purpose A/D converters often have control signals for simplified microprocessor interfacing. These ICs are appropriate for many process control, industrial, and instrumentation applications, as well as for environmental monitoring such as seismology, oceanography, meteorology, and pollution monitoring.

High-speed A/D converters have conversion times typically ranging from 400 ns to 3 μs. The higher speed performance of these devices is achieved by using the successive approximation technique, modified flash techniques, and statistically derived A/D conversion techniques. Applications appropriate for these

© 2006 by Taylor & Francis Group, LLC

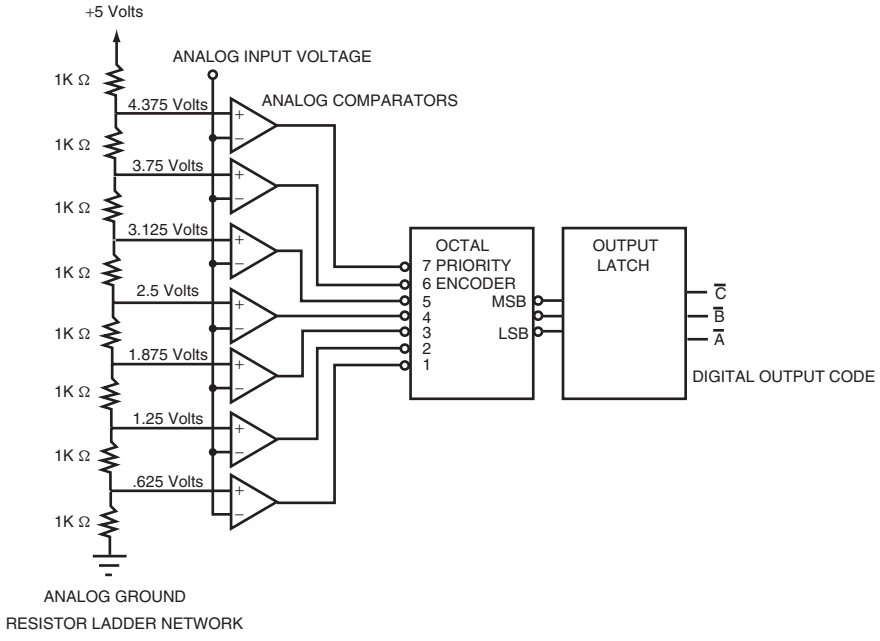


FIGURE 10.2 Flash A/D converter block diagram. (*Source:* Garrod, S. and Borns, R. *Digital Logic: Analysis, Application, and Design*, p. 928. Copyright ©1991 by Saunders College Publishing, Philadelphia, PA. Reprinted by permission of the publisher.)

A/D ICs include fast Fourier transform (FFT) analysis, radar digitization, medical instrumentation, and multiplexed data acquisition. Some ICs have been manufactured with an extremely high degree of linearity, to be appropriate for specialized applications in digital spectrum analysis, vibration analysis, geological research, sonar digitizing, and medical imaging.

Flash A/D converters have conversion times ranging typically from 10 to 50 ns. Flash A/D conversion techniques enable these ICs to be used in many specialized high-speed data acquisition applications such as TV video digitizing (encoding), radar analysis, transient analysis, high-speed digital oscilloscopes, medical ultrasound imaging, high-energy physics, and robotic vision applications.

Sampling A/D converters have a sample-and-hold amplifier circuit built into the IC. This eliminates the need for an external sample-and-hold circuit. The throughput of these A/D converter ICs ranges typically from 35 kHz to 100 MHz. The speed of the system is dependent on the A/D technique used by the sampling A/D converter.

A/D converter ICs produce digital codes in a serial or parallel format, and some ICs offer the designer both formats. The digital outputs are compatible with standard logic families to facilitate interfacing to other digital systems. In addition, some A/D converter ICs have a built-in analog multiplexer and therefore can accept more than one analog input signal.

Pulse code modulation (PCM) ICs are high-precision A/D converters. The PCM IC is often refered to as a PCM *codec* with both encoder and decoder functions. The encoder portion of the codec performs the A/D conversion, and the decoder portion of the codec performs the D/A conversion. The digital code is usually formatted as a serial data stream for ease of interfacing to digital transmission and multiplexing systems.

PCM is a technique where an analog signal is sampled, quantized, and then encoded as a digital word. The PCM IC can include successive approximation techniques or other techniques to accomplish the PCM encoding. In addition, the PCM codec may employ nonlinear data compression techniques, such as companding, if it is necessary to minimize the number of bits in the output digital code. Companding is a logarithmic technique used to compress a code to fewer bits before transmission. The inverse logarithmic function is then used to expand the code to its original number of bits before converting it to the analog

© 2006 by Taylor & Francis Group, LLC

signal. Companding is typically used in telecommunications transmission systems to minimize data transmission rates without degrading the resolution of low-amplitude signals. Two standardized companding techniques are used extensively: A-law and μ-law. The A-law companding is used in Europe, whereas the μ-law is used predominantly in the United States and Japan. Linear PCM conversion is used in high-fidelity audio systems to preserve the integrity of the audio signal throughout the entire analog range.

Digital signal processing (DSP) techniques provide another type of A/D conversion ICs. Specialized A/D conversion such as *adaptive differential pulse code modulation* (ADPCM), sigma–delta modulation, *speech subband encoding, adaptive predictive speech encoding*, and *speech recognition* can be accomplished through the use of DSP systems. Some DSP systems require analog front ends that employ traditional PCM codec ICs or DSP interface ICs. These ICs can interface to a digital signal processor for advanced A/D applications. Some manufacturers have incorporated DSP techniques on board the single-chip A/D IC, as in the case of the DSP56ACD16 sigma–delta modulation IC by Motorola.

Integrating A/D converters are used for conversions that must take place over a long period of time, such as digital voltmeter applications or sensor applications such as thermocouples. The integrating A/D converter produces a digital code that represents the average of the signal over time. Noise is reduced by means of the signal averaging, or integration. Dual-slope integration is accomplished by a counter that advances while an input voltage charges a capacitor in a specified time interval, T. This is compared to another count sequence that advances while a reference voltage is discharging across the same capacitor in a time interval, *delta t*. The ratio of the charging count value to the discharging count value is proportional to the ratio of the input voltage to the reference voltage. Hence, the integrating converter provides a digital code that is a measure of the input voltage averaged over time. The conversion accuracy is independent of the capacitor and the clock frequency since they affect both the charging and discharging operations. The charging period, T, is selected to be the period of the fundamental frequency to be rejected. The maximum conversion rate is slightly less than $1/(2\,\mathrm{T})$ conversions per second. While this limits the conversion rate to be too slow for high-speed data acquisition applications, it is appropriate for long-duration applications of slowly varying input signals.

Grounding and Bypassing on D/A and A/D ICs

D/A and A/D converter ICs require correct grounding and capacitive bypassing in order to operate according to performance specifications. The digital signals can severely impair analog signals. To combat the electromagnetic interference induced by the digital signals, the analog and digital grounds should be kept separate and should have only one common point on the circuit board. If possible, this common point should be the connection to the power supply.

Bypass capacitors are required at the power connections to the IC, the reference signal inputs, and the analog inputs to minimize noise that is induced by the digital signals. Each manufacturer specifies the recommended bypass capacitor locations and values in the data sheet. The manufacturers' recommendations should be followed to ensure proper performance.

Selection Criteria for D/A and A/D Converter ICs

Hundreds of D/A and A/D converter ICs are available, with prices ranging from a few dollars to several hundred dollars each. The selection of the appropriate type of converter is based on the application requirements of the system, the performance requirements, and cost. The following issues should be considered in order to select the appropriate converter.

1. What are the input and output requirements of the system? Specify all signal current and voltage ranges, logic levels, input and output impedances, digital codes, data rates, and data formats.
2. What level of accuracy is required? Determine the resolution needed throughout the analog voltage range, the dynamic response, the degree of linearity, and the number of bits encoding.
3. What speed is required? Determine the maximum analog input frequency for sampling in an A/D system, the number of bits for encoding each analog signal, and the rate of change of input digital codes in a D/A system.

© 2006 by Taylor & Francis Group, LLC

4. What is the operating environment of the system? Obtain information on the temperature range and power supply to select a converter that is accurate over the operating range.

Final selection of D/A and A/D converter ICs should be made by consulting manufacturers to obtain their technical specifications of the devices.

Defining Terms

Companding: A process designed to minimize the transmission bit rate of a signal by compressing it prior to transmission and expanding it upon reception. It is a rudimentary "data compression" technique that requires minimal processing.

Delta modulation: An A/D conversion process where the digital output code represents the change, or slope, of the analog input signal, rather than the absolute value of the analog input signal. A 1 indicates a rising slope of the input signal. A 0 indicates a falling slope of the input signal. The sampling rate is dependent on the derivative of the signal, since a rapidly changing signal would require a rapid sampling rate for acceptable performance.

Fixed reference D/A converter: The analog output is proportional to a fixed (nonvarying) reference signal.

Flash A/D: The fastest A/D conversion process available to date, also referred to as parallel A/D conversion. The analog signal is simultaneously evaluated by $2^n - 1$ comparators to produce an n-bit digital code in one step. Because of the large number of comparators required, the circuitry for flash A/D converters can be very expensive. This technique is commonly used in digital video systems.

Integrating A/D: The analog input signal is integrated over time to produce a digital signal that represents the area under the curve, or the integral.

Multiplying D/A: A D/A conversion process where the output signal is the product of a digital code multiplied times an analog input reference signal. This allows the analog reference signal to be scaled by a digital code.

Nyquist A/D converters: A/D converters that sample analog signals that have a maximum frequency that is less than the Nyquist frequency. The Nyquist frequency is defined as one-half of the sampling frequency. If a signal has frequencies above the Nyquist frequency, a distortion called *aliasing* occurs. To prevent aliasing, an *antialiasing filter* with a flat passband and very sharp rolloff is required.

Oversampling converters: A/D converters that sample frequencies at a rate much higher than the Nyquist frequency. Typical oversampling rates are 32 and 64 times the sampling rate that would be required with the Nyquist converters.

Pulse code modulation (PCM): An A/D conversion process requiring three steps: the analog signal is sampled, quantized, and encoded into a fixed length digital code. This technique is used in many digital voice and audio systems. The reverse process reconstructs an analog signal from the PCM code. The operation is very similar to other A/D techniques, but specific PCM circuits are optimized for the particular voice or audio application.

Sigma–delta A/D conversion: An *oversampling* A/D conversion process where the analog signal is sampled at rates much higher (typically 64 times) than the sampling rates that would be required with a Nyquist converter. Sigma–delta modulators integrate the analog signal before performing the delta modulation. The integral of the analog signal is encoded rather than the change in the analog signal, as is the case for traditional delta modulation. A digital sample rate reduction filter (also called a digital decimation filter) is used to provide an output sampling rate at twice the Nyquist frequency of the signal. The overall result of oversampling and digital sample rate reduction is greater resolution and less distortion compared to a Nyquist converter process.

Successive approximation: An A/D conversion process that systematically evaluates the analog signal in n steps to produce an n-bit digital code. The analog signal is successively compared to determine the digital code, beginning with the determination of the most significant bit of the code.

© 2006 by Taylor & Francis Group, LLC

References

Analog Devices. 1989. *Analog Devices Data Conversion Products Data Book*. Analog Devices, Inc., Norwood, MA.

Burr–Brown. 1989. *Burr-Brown Integrated Circuits Data Book*. Burr-Brown, Tucson, AZ.

DATEL. 1988. *DATEL Data Conversion Catalog*. DATEL, Inc., Mansfield, MA.

Drachler, W. and Bill, M. 1995. New High-Speed, Low-Power Data-Acquisition ICs. *Analog Dialogue* 29(2):3–6. Analog Devices, Inc., Norwood, MA.

Garrod, S. and Borns, R. 1991. *Digital Logic: Analysis, Application and Design,* Chap. 16. Saunders College Publishing, Philadelphia, PA.

Jacob, J.M. 1989. *Industrial Control Electronics*, Chap. 6. Prentice-Hall, Englewood Cliffs, NJ.

Keiser, B. and Strange, E. 1995. *Digital Telephony and Network Integration,* 2nd ed. Van Nostrand Reinhold, New York.

Motorola. 1989. *Motorola Telecommunications Data Book*. Motorola, Inc., Phoenix, AZ.

National Semiconductor. 1989. *National Semiconductor Data Acquisition Linear Devices Data Book*. National Semiconductor Corp., Santa Clara, CA.

Park, S. 1990. *Principles of Sigma-Delta Modulation for Analog-to-Digital Converters*. Motorola, Inc., Phoenix, AZ.

Texas Instruments. 1986. *Texas Instruments Digital Signal Processing Applications with the TMS320 Family*. Texas Instruments, Dallas, TX.

Texas Instruments. 1989. *Texas Instruments Linear Circuits Data Acquisition and Conversion Data Book*. Texas Instruments, Dallas, TX.

Further Information

Analog Devices, Inc. has edited or published several technical handbooks to assist design engineers with their data acquisition system requirements. These references should be consulted for extensive technical information and depth. The publications include *Analog-Digital Conversion Handbook*, by the engineering staff of Analog Devices, published by Prentice-Hall, Englewood Cliffs, NJ, 1986; *Nonlinear Circuits Handbook, Transducer Interfacing Handbook,* and *Synchro and Resolver Conversion*, all published by Analog Devices Inc., Norwood, MA.

Engineering trade journals and design publications often have articles describing recent A/D and D/A circuits and their applications. These publications include *EDN Magazine, EE Times*, and *IEEE Spectrum*. Research-related topics are covered in *IEEE Transactions on Circuits and Systems* and also *IEEE Transactions on Instrumentation and Measurement.*

© 2006 by Taylor & Francis Group, LLC

11

Application-Specific Integrated Circuits

Constantine N. Anagnostopoulos

Paul P.K. Lee

11.1 Introduction **11**-1
11.2 Full Custom ASICs **11**-2
11.3 Semicustom ASICs **11**-3
Gate Arrays • Standard Cells • Functional Blocks • Analog Arrays

11.1 Introduction

Application specific integrated circuits (**ASICs**), also called custom ICs, are chips specially designed to (1) perform a function that cannot be done using standard components, (2) improve the performance of a circuit, or (3) reduce the volume, weight, and power requirement and increase the reliability of a given system by integrating a large number of functions on a single chip or a small number of chips.

ASICs can be classified into the following three categories: (1) full custom, (2) semicustom, and (3) programmable logic devices (PLDs).

The first step of the process toward realizing a custom IC chip is to define the function the chip must perform. This is accomplished during system partitioning, at which time the system engineers and IC designers make some initial decisions as to which circuit functions will be implemented using standard, off the shelf, components and which will require custom ICs. After several iterations the functions that each custom chip has to perform are determined.

There usually are many different ways by which a given function may be implemented. For example, the function may be performed in either the analog or the digital domain. If a digital approach is selected, different execution strategies may be chosen. For example, a delay function may be implemented either by a shift register or by a random access memory. Thus, in the second step of the process, the system engineers and IC designers decide how the functions each of the chips has to perform will be executed. How a function is executed by a given chip constitutes the behavioral description of that chip.

Next, a design approach needs to be developed for each of the custom chips. Depending on the implementation method selected, either a full custom, a semicustom, or a user programmable device approach is chosen.

One important consideration in the design choice is cost and turn around time (Fey and Paraskevopoulos, 1985). Typically, a full custom approach takes the longest to design, and the cost per chip is very high unless the chip volume is also high, normally more than a few hundred thousands chips per year. The shortest

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

design approach is using programmable devices but the cost per chip is the highest. This approach is best for prototype and limited production systems.

11.2 Full Custom ASICs

In a typical full custom IC design, every device and circuit element on the chip is designed for that particular chip. Of course, common sense dictates that device and circuit elements proven to work well in previous designs are reused in subsequent ones whenever possible.

A full custom approach is selected to minimize the chip size or to implement a function that is not available or would not be optimum with semicustom or standard ICs. Minimizing chip size increases the fabrication yield and the number of chips per wafer. Both of these factors tend to reduce the cost per chip.

Fabrication yield is given by

$$Y = [(1 - e^{-AD})/AD]^2$$

where A is the chip or die area and D is the average number of defects per square centimeter per wafer. The number of die per wafer is given by

$$N = [\pi(R - A^{1/2})^2]/A$$

where R is the wafer radius and A is, again, the area of the die.

Additional costs incurred involve testing of the die while still on the wafer and packaging a number of the good ones and testing them again. Then good die must be subjected to reliability testing to find out the expected lifetime of the chip at normal operating conditions (Hu, 1992).

Fabrication of full custom ASICs is done at silicon foundries. Some of the foundries are captive, that is, they fabricate devices only for the system divisions of their own company. Others make available some of their lines to outside customers. There also are foundries that exclusively serve external customers. Foundries often provide design services as well. If users are interested in doing their own design, however, then the foundry provides a set of design rules for each of the processes they have available. The design rules describe in broad terms the fabrication technology, whether CMOS, bipolar, BiCMOS, or GaAs and then specify in detail the minimum dimensions that can be defined on the wafer for the various layers, the SPICE [Vladimirescu et al. 1981] parameters of each of the active devices, the range of values for the passive devices and other rules and limitations.

Designing a full custom chip is a complex task and can only be done by expert IC designers. Often a team of people is required both to reduce the design time (Fey and Paraskevopoulos, 1986) and because one person may simply not have all the design expertise required. Also, sophisticated and powerful computer hardware and software are needed. Typically, the more sophisticated the computer aided design (CAD) tools are, the higher the probability that the chip will work the first time. Given the long design and fabrication cycle for a full custom chip and its high cost, it is important that as much as possible of the design be automated, that design rule checking should be utilized, and that the circuit simulation be as complete and accurate as possible.

During the design phase, a substantial effort should be made to incorporate on the chip additional circuitry to help verify that the chip is working properly after it is fabricated, or to help identify the section or small circuit responsible for the chip not working, or verify that an error occurred during the fabrication process. For digital full custom circuits, a large number of testability techniques have been developed (Williams and Mercer, 1993), most of which are automated and easily incorporated in the design, given the proper software. For analog circuits there are no accepted universal testability methods. A unique test methodology for each circuit needs to be developed.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Technology selection for a particular full custom ASIC depends on the functions the chip has to perform, the performance specifications, and its desired cost.

11.3 Semicustom ASICs

The main distinguishing feature of semicustom ASICs, compared to full custom ones, is that the basic circuit building blocks, whether analog or digital, are already designed and proven to work. These basic circuits typically reside in libraries within a CAD system. The users simply select from the library the components needed, place them on their circuits, and interconnect them. Circuit simulation is also done at a much higher level than SPICE, and the designer is, therefore, not required to be familiar with either semiconductor or device physics.

Semicustom ICs are designed using either **gate arrays, standard cells**, analog arrays, **functional blocks** and PLDs such as field programmable gate arrays (FPGAs). Note that there does not exist a standard naming convention for these products within the industry. Often different manufacturers use different names to describe what are essentially very similar products. Gate arrays, standard cells, and FPGAs are used for digital designs. Analog arrays are used for analog designs and functional blocks are used for both.

Gate Arrays

A gate array consists of a regular array of transistors, usually arranged in two pairs of n- and p-channel, which is the minimum number required to form a NAND gate, and a fixed number of bonding pads, each incorporating an input/output (I/O) buffer. The major distinguishing feature of gate arrays is that they are partially prefabricated by the manufacturer. The designer customizes only the final contact and metal layers. Partially prefabricating the devices reduces delivery time and cost, particularly of prototype parts.

The layout of a simple 2048-gates CMOS gate array is shown in Fig. 11.1. The device consists of 16 columns of transistors and each column contains 128 pairs of n-channel and p-channel transistors. Between the columns are 18 vertical wiring channels, each containing 21 tracks. There are no active devices in the channels. There are 4 horizontal wiring tracks for each gate for a total of 512 horizontal tracks or *routes* for the whole array. The optimum number R of wiring tracks or routes per gate, of an array of a given number of gates, is given by the empirical formula (Fier and Heikkila, 1982)

$$R = 3CG^{0.124}$$

Where C is the average number of connections per gate and G is the number of gates in the array. For a two-input NAND gate, shown in Fig. 11.2, the number of connections C is 3, the two inputs A and B and the output AB, and for a 2048-gates array, the preceding formula gives $R = 23$. In this device 25 routes per gate are provided.

In the perimeter of the device 68 bond pads are arranged. Of these, 8 pads are needed for power and ground connections. Another empirical formula (Fier and Heikkila, 1982), based on Rent's rule, specifies that the number of I/O pads required to communicate effectively with the internal gates is given by

$$P = CG^{a}$$

where P is the sum of input and output pads, C is again the number of connections per gate, G is the number of gates in the array and a is Rent's exponent having a value between 0.5 and 0.7. For large-scale integrated (LSI) circuits, typically $a = 0.46$. Assuming this value for a, then P is equal to 134 for $G = 2048$ and $C = 3$. This gate array, therefore, with only 60 I/O pads should be pad limited for many designs.

As mentioned previously, the process of designing with gate arrays begins with the designer drawing, with the aid of a CAD system, the circuit schematic representing the function the chip must perform. This activity is called **schematic capture**. The schematic, containing circuit elements or *cells*, such as inverters, NAND and NOR gates, flip-flops, adders, etc., is formed by getting these components from a specified

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

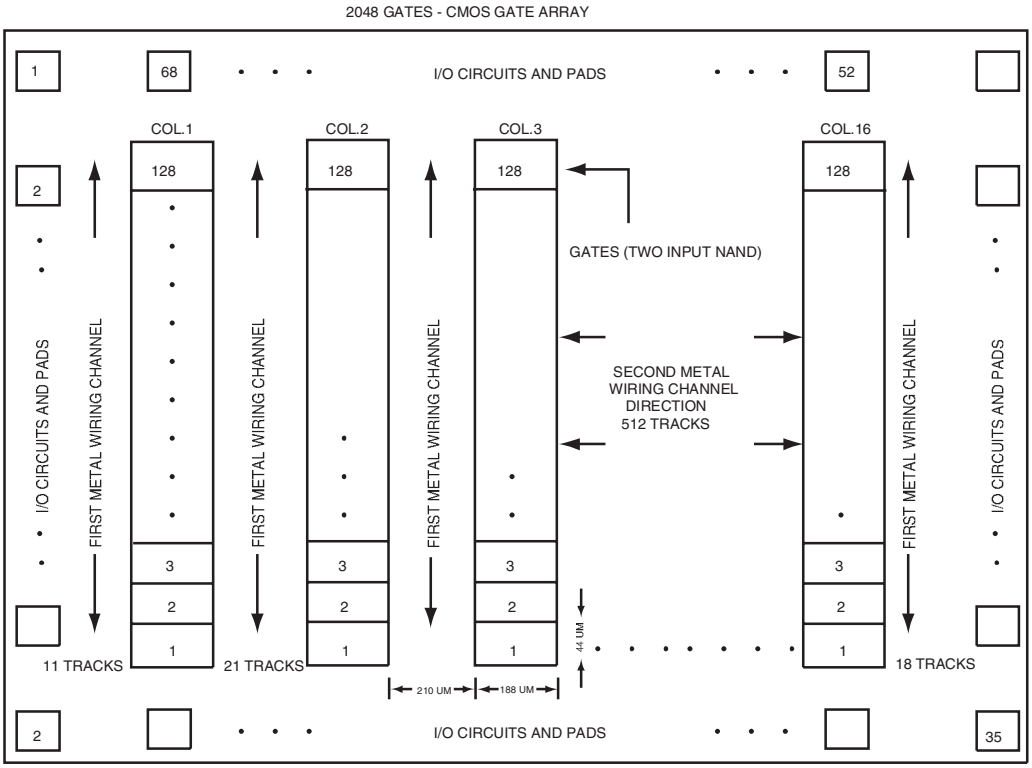


FIGURE 11.1 Schematic layout of a 2048-gates CMOS gate array.

library in the computer. Each of the cells has a number of representations associated with it. It has a schematic used, of course, in drawing up the complete circuit schematic. It has a functional description that specifies what the element does, for example, an inverter takes an 1 input and produces a 0 in its output and vice versa. Another file describes its electrical characteristics, such as the time delay between when the inputs reach a certain state and the time the output responds to that state, often referred to as the propagation delay, and it has a physical representation.

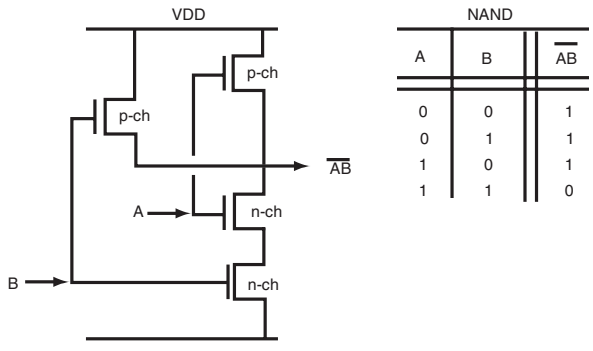


FIGURE 11.2 Electrical schematic of a CMOS NAND logic gate and its truth table.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

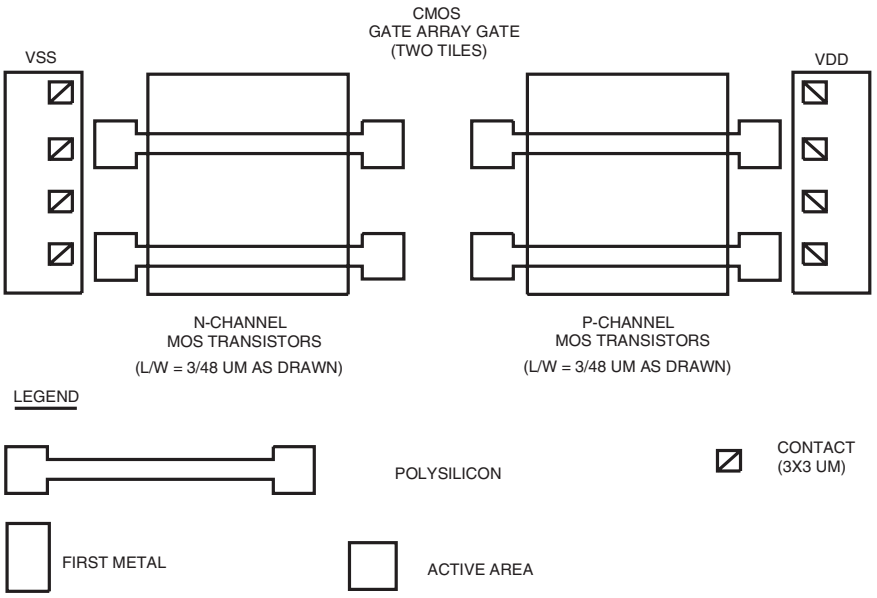


FIGURE 11.3 Schematic layout of one of the 2048 uncommitted gates in a device column.

In the array shown in Fig. 11.1, before the customized layers are placed on the device, each of the 2048 gates appears schematically as shown in Fig. 11.3. There are two n-channel transistors on the left and two p-channel transistors on the right. There is also a ground or VSS buss line running vertically on the left and a VDD line on the right. Figure 11.4 shows the equivalent electrical schematic of the structure in Fig. 11.3. In Fig. 11.5 is shown only one pair of n- and p-channel transistors and locations within the device area where one of the customizing layers, the contacts, may be placed. The physical representation of a particular library cell contains information of how the uncommitted transistors in the device columns should be connected.

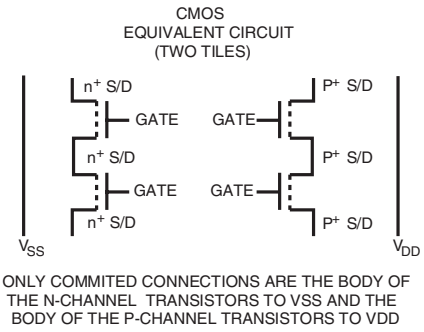


FIGURE 11.4 Equivalent electrical schematic of an uncommitted gate in the device column area of the gate array.

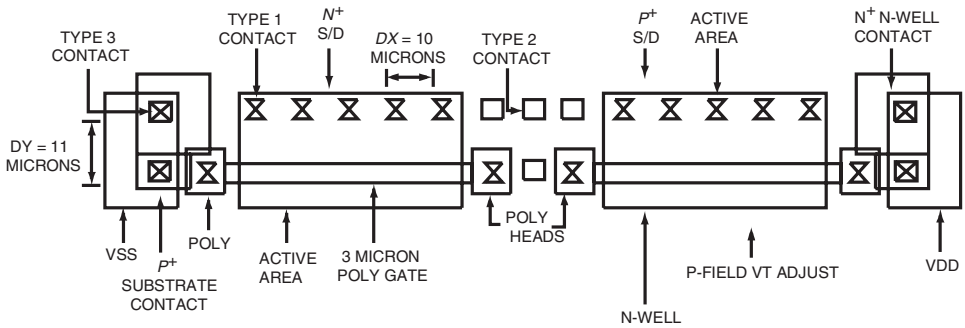


FIGURE 11.5 Layout of a single pair of n- and p-channel transistors in the device column showing how the contact locations are allocated.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

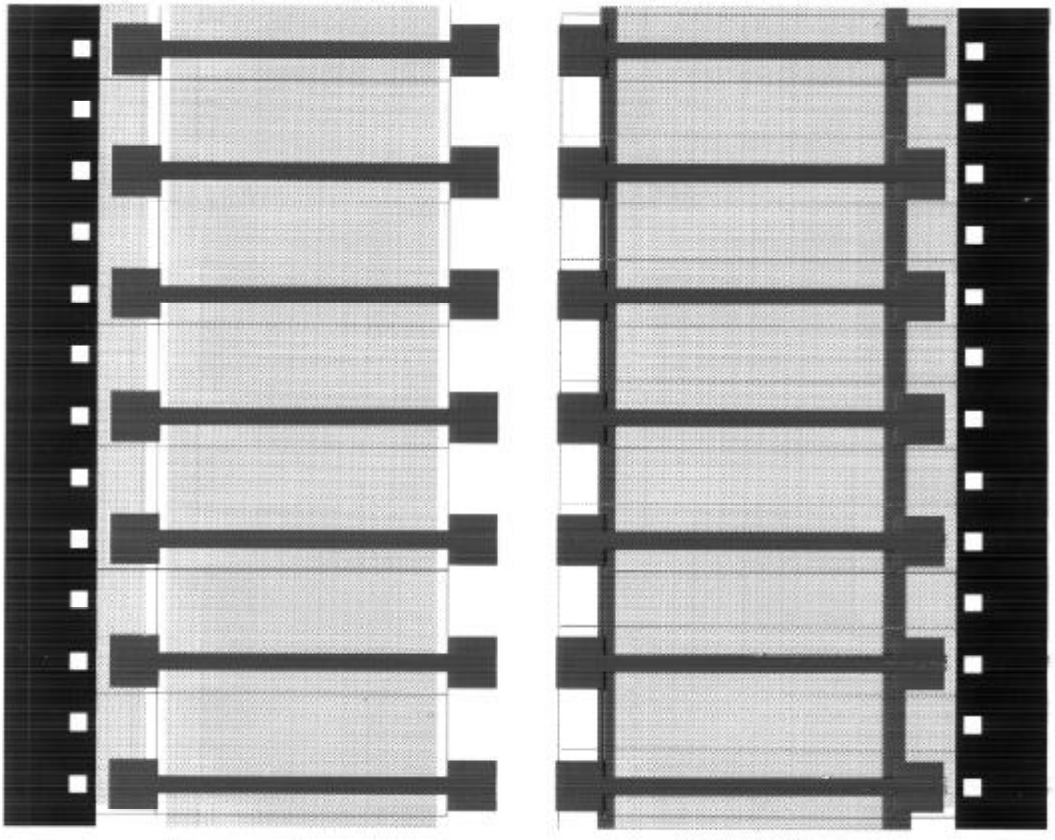


FIGURE 11.6 Layout of a portion of a device column.

As an example, in Fig. 11.6 is shown a portion of one of the 16 device columns in the array. Figure 11.7 shows the location of the contact holes, shown as white squares, and the metal interconnects, for an inverter shown as the dark horizontal bars. An actual inverter circuit is realized by placing this inverter cell along anywhere in the column, as shown in Fig. 11.8. Figure 11.9 shows the electrical connections made in the gate array and in Fig. 11.9(b) is shown, for reference, the electric schematic for an inverter. Note that in Fig. 11.8 and Fig. 11.9, the polysilicon gates of the top and bottom n-channel transistors are connected to VSS and the corresponding gates of the the p-channel transistors are connected to VDD. This is done to isolate the inverter from interfering electrically with another logic gate that may be placed above or below it.

Returning to the design process, after schematic capture is completed, the designer simulates the entire circuit first to verify that it performs the logic functions the circuit is designed to perform and then using the electrical specifications files, a timing simulation is performed to make sure the circuit will operate at the clock frequency desired. However, the timing simulation is not yet complete. In the electrical characteristics file of each cell, a typical input and load capacitance is assumed. In an actual circuit the load capacitance may be quite different. Therefore, another timing simulation must be done after the actual capacitance values are found for each node.

To do this, first from the circuit schematic, a *netlist* is extracted. This list contains exact information about which node of a given element is connected to which nodes of other elements. This netlist is then submitted to the *place and route* program. This program places all of the elements or cells in the circuit on the gate array, in a fashion similar to what was done in Fig. 11.8, and attempts to connect them as specified in the netlist. The program may iterate the placement of cells and interconnection or routing until all cells listed in the netlist are placed on the gate array and interconnected as specified.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

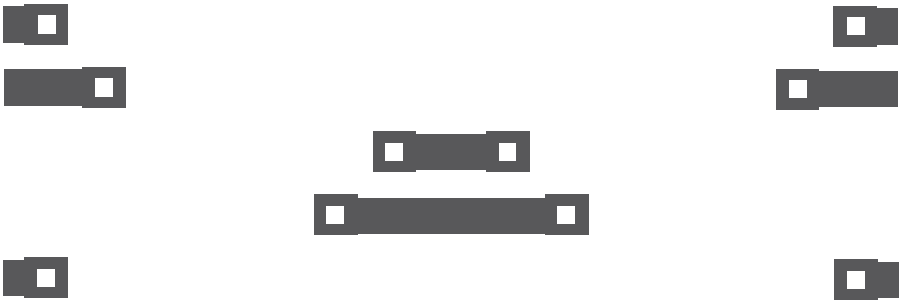


FIGURE 11.7 Layout of a single inverter library cell.

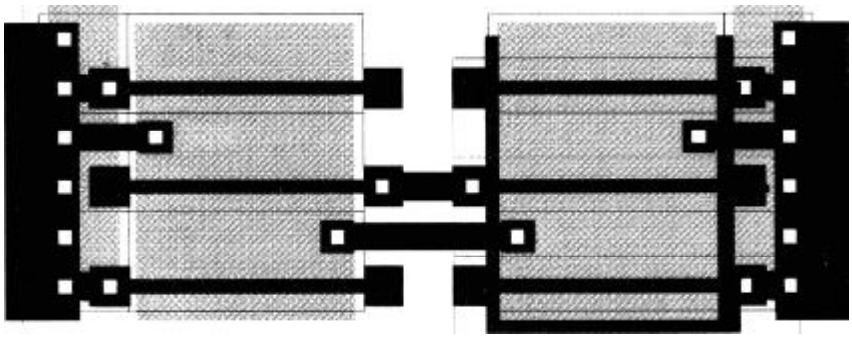
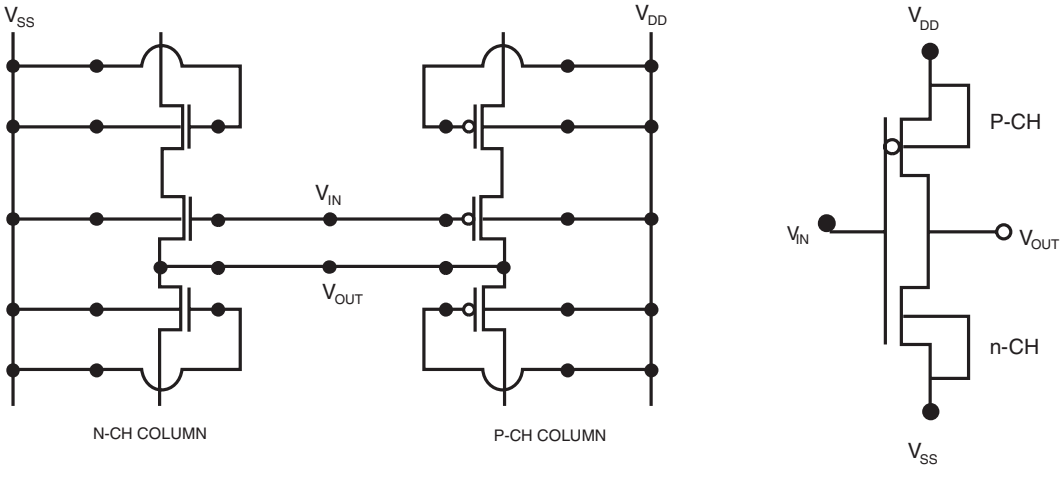


FIGURE 11.8 Completed layout of an inverter in device column.



(a) PLACED INVERTER ELECTRICAL CONNECTIONS (b) INVERTER TYPICAL SCHEMATIC

FIGURE 11.9 (a) Electrical connections of a placed inverter in a device column, (b) electrical schematic of an inverter.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

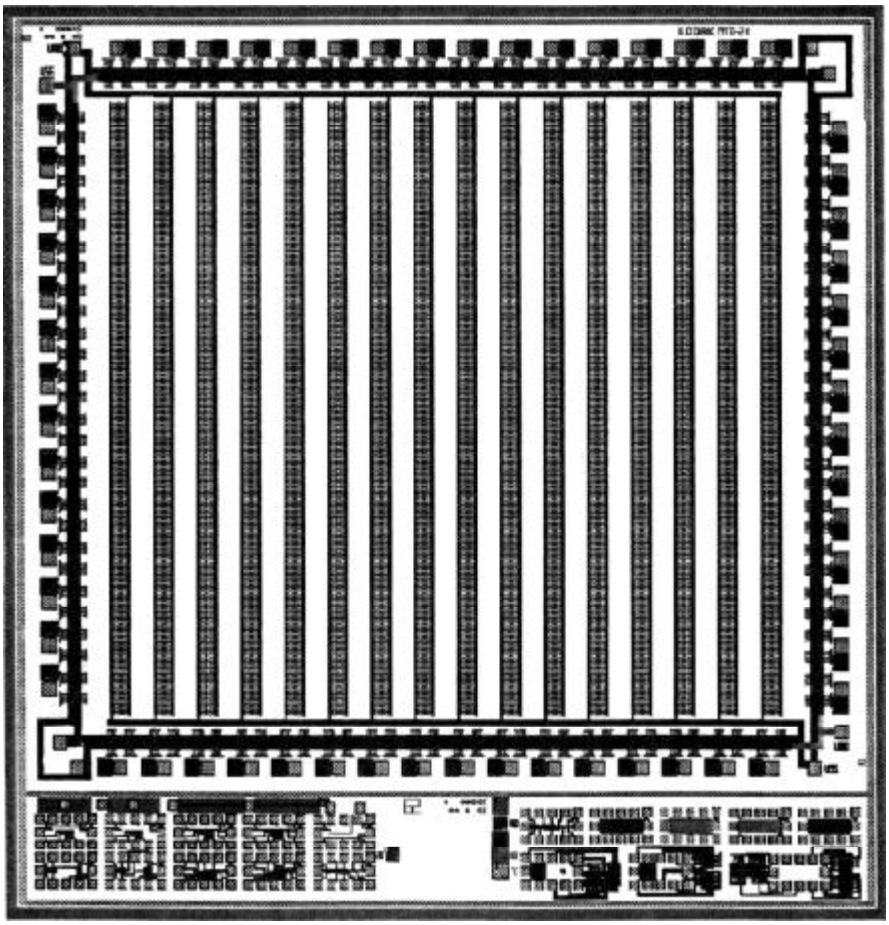


FIGURE 11.10 Layout of the actual 2048-gates gate array before customization.

The routing portion of the place and route program is called a *channel router*. It derives its name by the method it uses to interconnect the cells in the array. Also, the gate array architecture, shown in Fig. 11.1, with its vertical wiring channels and a single horizontal wiring channel, is selected so that this type of router can be used. Figure 11.10 shows the actual 2048-gates gate array, before customization and Fig. 11.11 shows after it is completed with a custom circuit. In Fig. 11.12 a corner of the array is shown in higher magnification. The vertical darker lines are the first level metal interconnects and the horizontal lighter lines are the second level metal interconnects. This array requires four layers to be customized. These are the first and second level metals, contact, and the via. Vias are contacts exclusively between the two metal layers. Note that all internal wiring in the cells, like the inverter in Fig. 11.7, are made with first level metal and, therefore, the router is not allowed to route first level metal over the device areas.

Once the circuit is placed and routed, an extraction program is applied that does two things. First, it calculates the actual resistance and capacitance seen at each node of the circuit that is then fed back into the netlist for more accurate timing simulation; second, it extracts a new netlist of the circuit on the gate array that can be compared with the initial netlist. The two netlists should, of course, be identical. However, often manual intervention is required to finish either the placement or routing of a particularly difficult circuit and that could cause an inadvertent error to occur in the layout. The error is then detected by comparing this extracted netlist to the original one.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

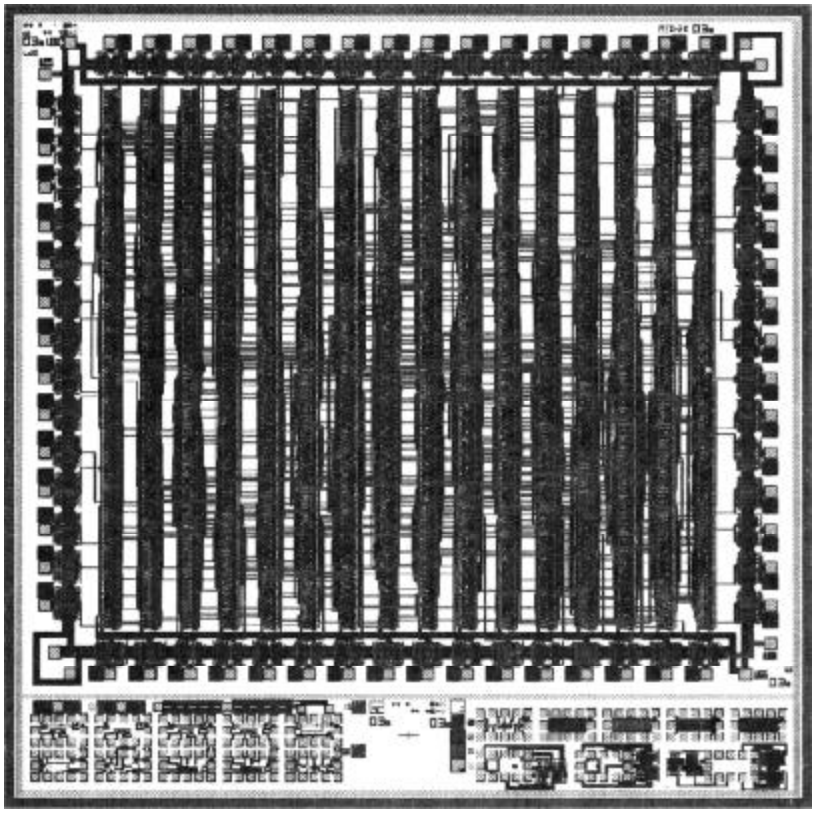


FIGURE 11.11 A custom circuit implementation on the 2048-gates gate array.

After the design is completed, **test vectors** must be generated. Two sets of test vectors are generated. The input set and the corresponding output set. Each test vector consists of a number of 1s and 0s. The input test vector has as many elements as the number of chip input pins and the output test vector, likewise, has as many elements as the number of output pins.

The input test vector is not, typically, similar to what the chip will see in actual operation. Instead, the goal is to select a set of input vectors that when applied to the chip will cause every internal node to change state at least once. The timing verification program is used for this task. The program keeps track of the nodes toggled, as each input vector is applied, and also saves the output vector. When the finished devices are received, the same input test vectors are applied in sequence, and the resulting output vectors are captured and compared to those obtained from the timing simulation runs. If the two match perfectly, the device is classified as good.

The purpose of this test is to detect whether, because of a defect during fabrication, an internal gate is not operating properly. Usually if a test can cause about 90% of the internal nodes to switch state, it is considered to provide adequate coverage. It would be desirable to have 100% coverage, but often this is not practical because of the length of time it would take for the test to be completed. Finally, because the test is done at the frequency at which the chip will operate, only minimal functional testing is performed.

Just as in the device columns any logic gate can be placed anywhere in any of the columns, each of the pads can serve as either an input, an output, or a power pad, again by customizing the four layers. Figure 11.13 shows an I/O buffer at the end of the prefabrication process. Figure 11.14 shows the contact and metal pattern to make the uncommitted pad an inverting output buffer and Fig. 11.15 shows the two

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

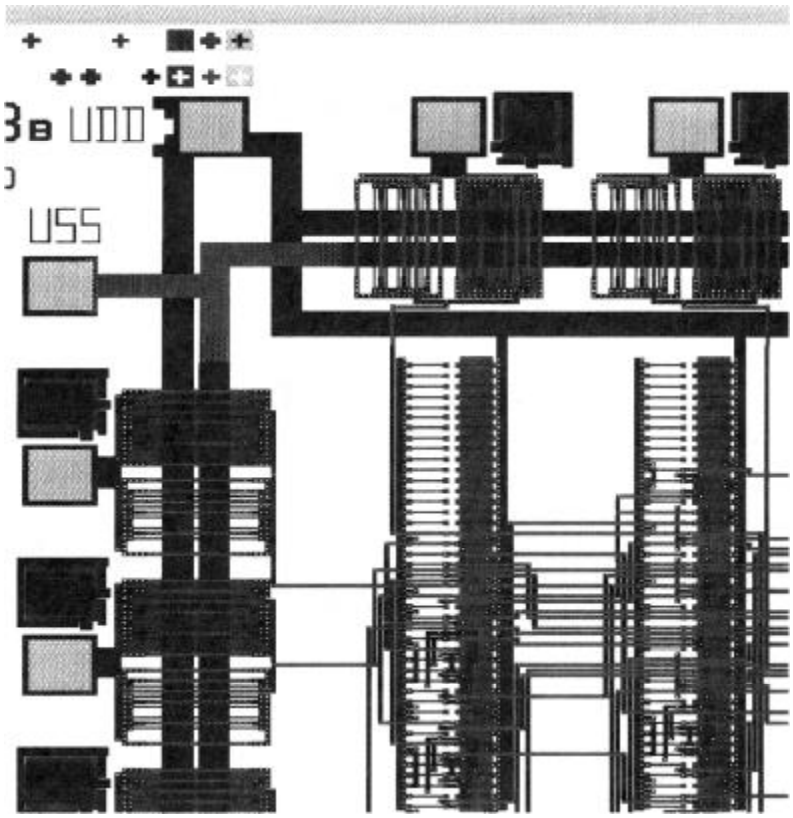


FIGURE 11.12 Closeup of a corner of the ASIC shown in Fig. 11.11.

together. Figure 11.16 shows the circuit schematic for the output buffer as well as the schematic when the pad is customized as an input buffer, the layout of which is shown in Fig. 11.17.

The gate array architecture presented here is not the only one. A number of different designs have been developed and are currently used. For each of the architectures, a different place and route algorithm is then developed to take advantage of that particular design. Gate arrays have also taken advantage of developments in silicon technology, which primarily consists of shrinking of the minimum dimensions. As a result, arrays with several million transistors per chip are presently in use.

Standard Cells

Designing with gate arrays is fairly straightforward and is utilized extensively by system engineers. However, gate arrays are quite inefficient in silicon area use since the number of transistors, the number of routing tracks, and the number of I/O pads needed by a particular circuit does not always match the number of devices or pads on the array, which are, of course, fixed in number.

The standard cells design methodology was developed to address this problem. As was the case for gate arrays, a predesigned and characterized library of cells exists for the circuit designer to use. But, unlike gate arrays, the chip is layed out containing only the number of transistors and I/O pads specified by the particular circuit, resulting in a smaller, less expensive chip. Fabrication of a standard cells circuit, however, must start from the beginning of the process. This has two major drawbacks. First, the fabrication time for standard cells is several weeks longer than for gate arrays; second, the customer has to pay for the fabrication of the whole lot of wafers, whereas for gate arrays, only a small portion of the lot is processed for each customer. This substantially increases the cost of prototype parts.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

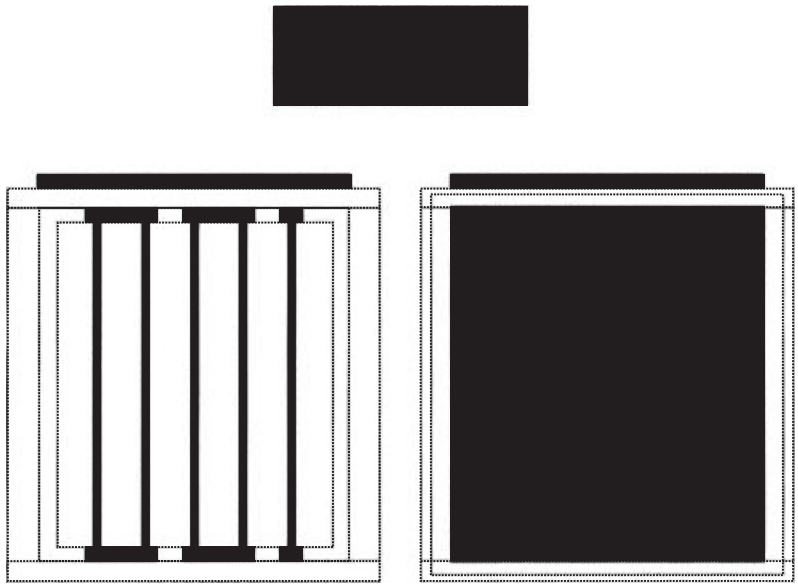


FIGURE 11.13 Layout of an uncommitted I/O buffer. In this figure, the bonding pad is the rectangular pattern at the top, the n-channel transistor is on the left and the p-channel is located on the right.

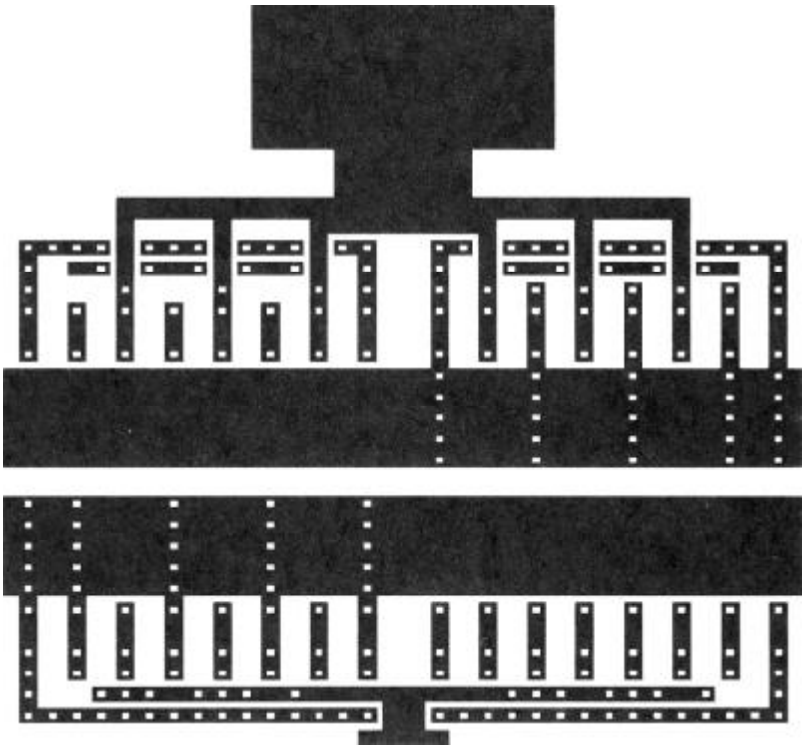


FIGURE 11.14 Layout of the contact and metal pattern of the output buffer library cell.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

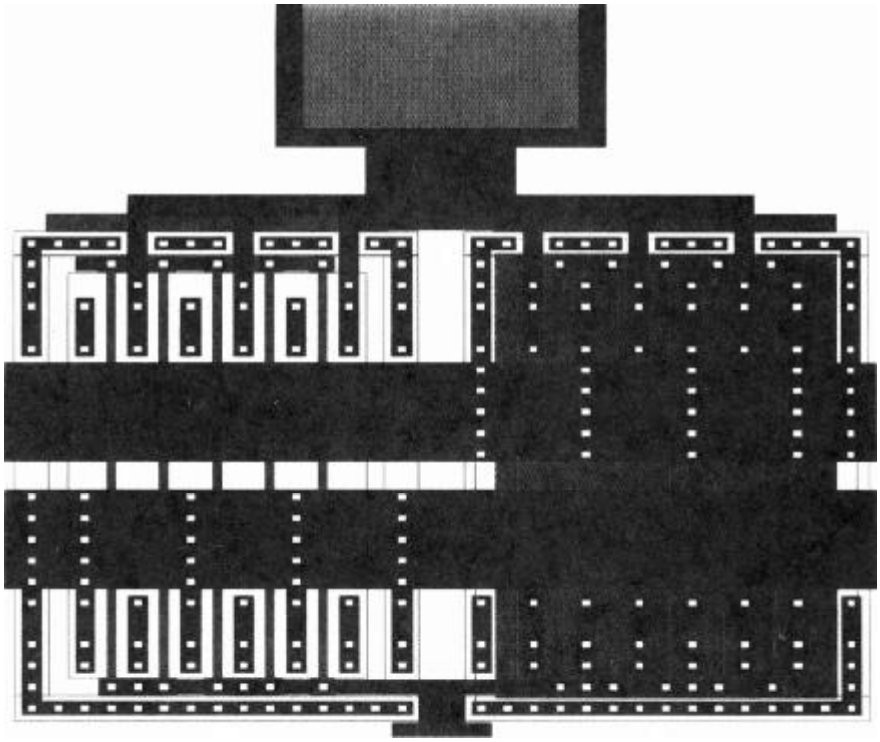


FIGURE 11.15 Completed layout of a full inverting output buffer.

Because standard cells are placed and routed automatically, the cells have to be designed with some restrictions, such as uniform height, preassigned locations for clocks and power lines, predefined input and output port locations, and others. Figure 11.18 shows the layout of three common cells from a standard cells library and Fig. 11.19 shows a completed standard cells chip. Its appearance is very similar to that of a gate array, but it contains no unused transistors or I/O pads and the minimum number of routing channels.

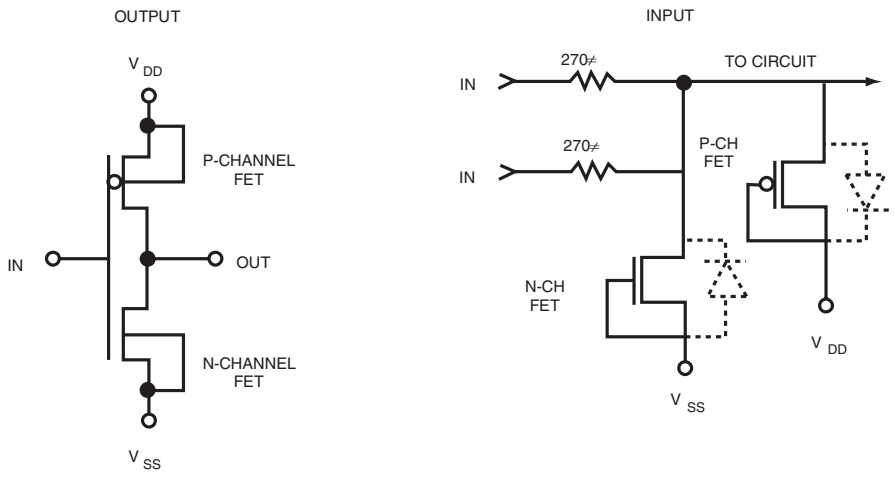


FIGURE 11.16 Electrical schematics of the input and output buffers. For TTL compatible output or input, or for other reasons, the buffers may use only some of the five segments of either the n- or p-channel transistors.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

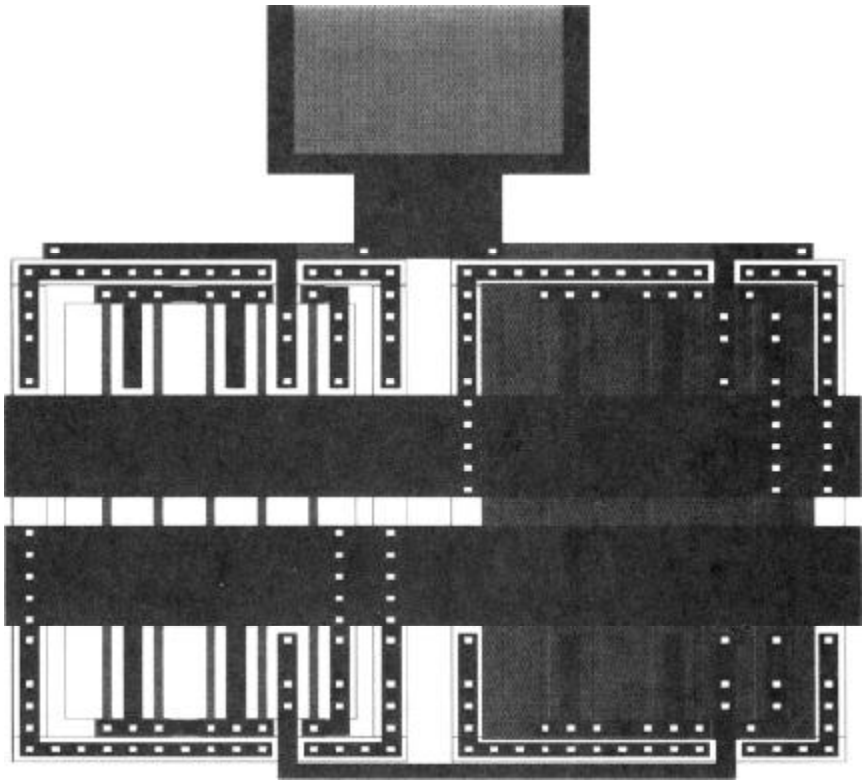


FIGURE 11.17 Completed layout of a simple noninverting input buffer.

Functional Blocks

Advances in place and route software have made possible the functional blocks design methodology. Here the cells are no longer simple logic gates, but complete functions. The cells can be analog or digital and can have arbitrary sizes. Figure 11.20 shows an example of a digital functional block design. In the figure, the three white blocks are placed and routed automatically along with several rows of standard cells. In Fig. 11.21, a small linear charge-coupled device (CCD) image sensor, located in the center of the chip, and a few other analog circuits, located below it, are placed on the same chip with a block of standard cells that *C* produce all of the logic needed to run the chip. In this chip, care is taken as to both the placement of the various blocks and the routing because of the sensitivity of the analog circuits to noise, temperature, and other factors. Therefore, the designer had considerable input as to the placement of the various blocks and the routing, and the software is designed to allow this intervention, when needed.

Often the digital functional blocks are built by a class of software programs called **silicon compilers** or **synthesis tools**. With such tools circuit designers typically describe the functionality of their circuit at the behavioral level rather than the gate level, using specially developed computer languages called *hardware description languages* (HDL). These tools can quickly build memory blocks of arbitrary size, adders or multipliers of any length, PLAs, and many other circuits.

It should be obvious by now that the semicustom design approach makes heavy use of software tools. These tools eliminate the tedious and error prone task of hand layout, provide accurate circuit simulation, and help with issues such as testability. Their power and sophistication make transparent the complexity inherent in the design of integrated circuits.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

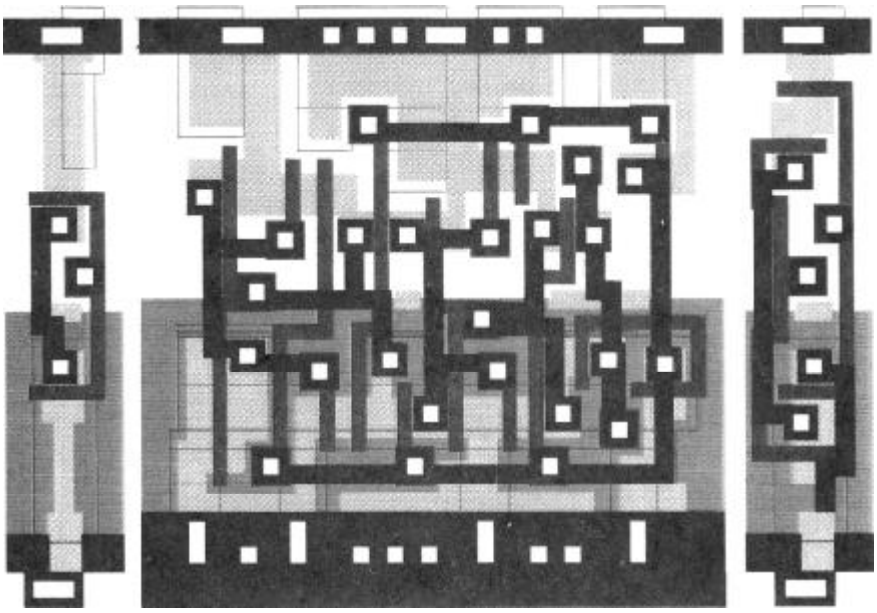


FIGURE 11.18 Three standard cells. A two-input NAND gate is shown on the left, a D type flip-flop is in the middle, and a simple inverter is shown on the right.

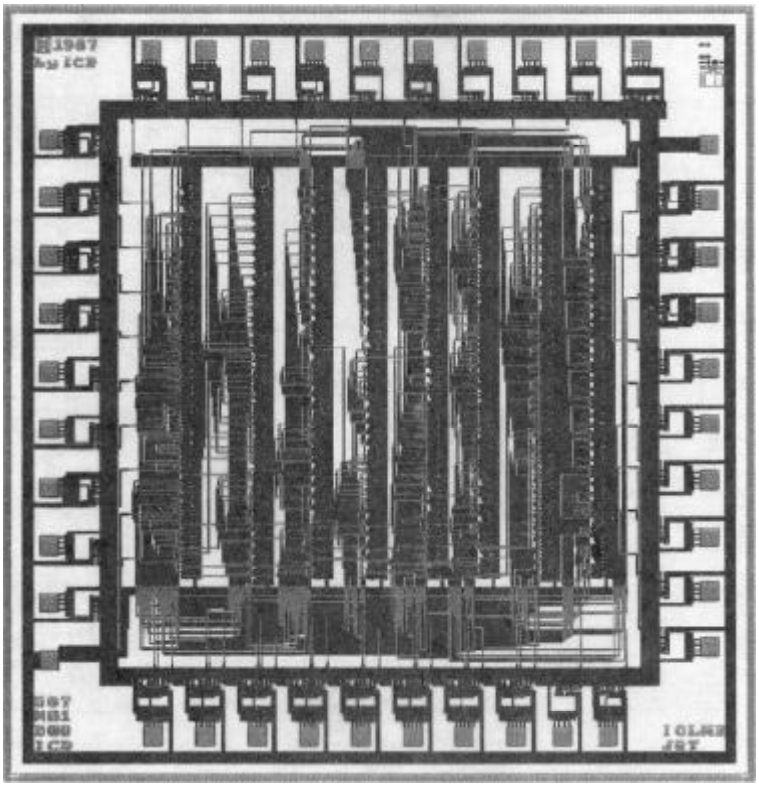


FIGURE 11.19 Layout of a completed standard cell ASIC.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

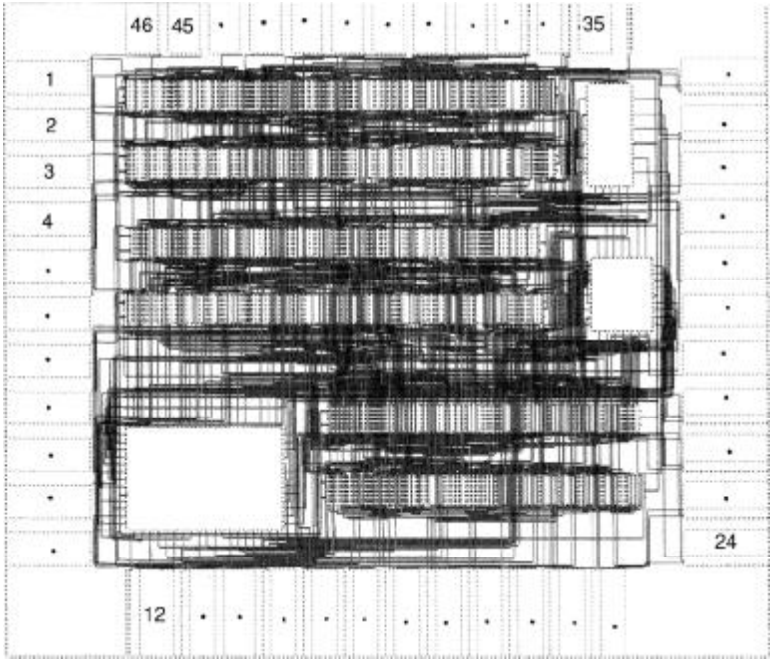


FIGURE 11.20 Layout of a digital functional block designed ASIC.

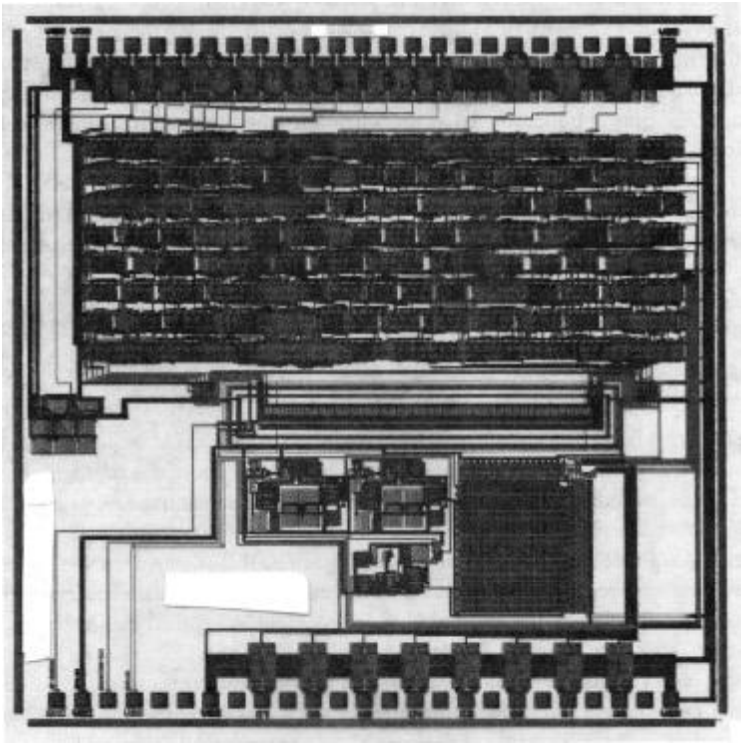


FIGURE 11.21 Layout of a mixed (combined analog and digital circuits on the same chip) functional block designed ASIC.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

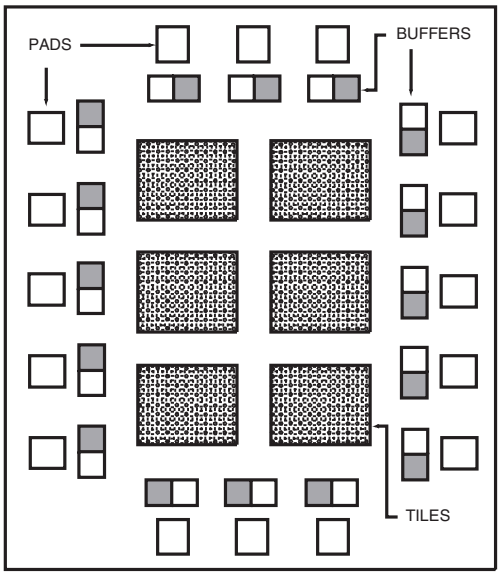


FIGURE 11.22 Architecture of an analog array.

Analog Arrays

Analog arrays are typically made in a bipolar process and are intended for the fabrication of high-performance analog circuits. Like gate arrays, these devices are prefabricated up to the contact level and, like all semicustom approaches, they are provided with a predesigned and characterized cell library. Because of the infinite variety of analog circuits, however, designers often design many of their own cells

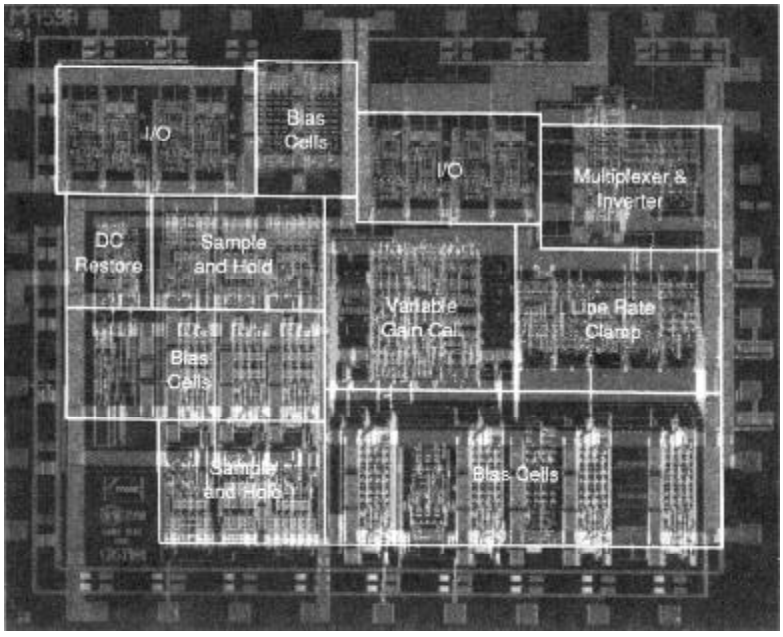


FIGURE 11.23 Example of an ASIC (Boisvert and Gaboury, 1992) implemented in a commercially available analog array.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

or modify the ones provided. Unlike digital circuits, the layout is done manually because analog circuits are more sensitive to temperature gradients, power supply voltage drops, crosstalk, and other factors and because the presently available software is not sufficiently sophisticated to take all of these factors into account.

The architecture of analog arrays is *tile* like, as shown in Fig. 11.22. Each tile is identical to all other tiles. Within each tile is contained a number of transistors, resistors, and capacitors of various sizes. The array also contains a fixed number of I/O pads, each of which can be customized with the last four layers to serve either as an input or output buffer. Figure 11.23 shows an actual circuit (Boisvert and Gaboury, 1992) implemented in a commercially available analog array.

Defining Terms

ASICs: The acronym of *application specific integrated circuits*; another name for such chips is *custom integrated circuits*.

Functional blocks: ASICs designed using this methodology are more compact than either gate arrays or standard cells because the blocks can perform much more complex functions than do simple logic gates.

Gate arrays: Chips that contain uncommitted arrays of transistors and are prefabricated up to a certain step after which they are customized for the particular application.

Nonreccurring engineering (NRE): Costs the foundry charges the ASIC customer. These costs include engineering time, the cost of making the masks, the cost of fabricating one lot of wafers, and for packaging and testing the prototype parts.

Standard cells: A design methodology for realizing ASICs. Compared to gate arrays, they make more efficient use of silicon.

Schematic capture: The process by which the functionality of the chip is captured in an electrical schematic, usually with the aid of a computer, and using components from a cell library resident in that computer.

Silicon compilers, synthesis tools: Software programs that can construct an ASIC whose functionality is no longer described by a circuit schematic but in a special high level computer languages, generally called hardware description languages or HDL.

Test vectors: A test scheme that consists of pairs of input and output. Each input vector is a unique set of 1s and 0s applied to the chip inputs and the corresponding output vector is the set of 1s and 0s produced at each of the chip's output.

References

Boisvert, D.M. and Gaboury, M.J. 1992. An 8–10-bit, 1–40 MHz analog signal processor with configurable performance for electronic imaging applications. In *Proceedings IEEE International ASIC Conference and Exhibit*, pp. 396–400. Rochester, NY.

Fey, C.F. and Paraskevopoulos, D. 1985. Selection of cost effective LSI design methodologies. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 148–153. Portland, OR.

Fey, C.F. and Paraskevopoulos, D. 1986. A model of design schedules for application specific ICs. In *Proceedings IEEE Custom Integrated Circuits Conference*, pp. 490–496. Rochester, NY.

Fier, D.F. and Heikkila, W.W. 1982. High performance CMOS design methodologies. In *Proceedings IEEE Custom Integrated Circuits Conference*, pp. 325–328. Rochester, NY.

Hu, C. 1992. IC reliability simulation. *IEEE J. of Solid State Circuits* 27(3):241–246; see also Proceedings of the Annual International Reliability Physics Symposium.

Ting, G., Guidash, R.M., Lee, P.P.K., and Anagnostopoulos, C. 1994. A low-cost, smart-power BiCMOS driver chip for medium power applications. In *Proceedings IEEE International ASIC Conference and Exhibit*, pp. 466–469. Rochester, NY.

Vladimirescu, A. et al. 1981. *SPICE Manual*. Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, Oct.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Williams, T.W. and Mercer, M.R. 1993. Testing digital circuits and design for testability. In *Proceedings IEEE International ASIC Conference and Exhibit* (Tutorial Session), p. 10. Rochester, NY.

Further Information

The ASIC field has been expanding rapidly since about 1980. The best source of current information is the two major conferences where the more recent technology developments are reported. These are the IEEE Custom Integrated Circuits Conference (CICC) held annually in May and the IEEE International ASIC Conference and Exhibit, held annually in September. *IEEE Spectrum Magazine* lists these conferences in the calender of events. Apart from the regular technical sessions, educational sessions and exhibits by ASIC vendors are part of these two conferences.

An additional resource is the USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292-6695, Telephone (213) 822-1511.

Finally, a selected number of papers from the CICC are published every year, since 1984, in special issues of the *IEEE Journal of Solid State Circuits*.

Additional information can also be found in a number of other conferences and their proceedings, including the Design Automation Conference (DAC) and the International Solid States Circuits Conference (ISSCC).

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

12 Digital Filters

12.1 Introduction **12**-1
12.2 FIR Filters **12**-1
Fundamentals • Structures • Design Techniques • Multirate and Adaptive FIR Filters • Applications
12.3 Infinite Impulse Response (IIR) Filters **12**-7
Realizations • IIR Filter Design • Analog Filters • Design Using Bilinear Transformations
12.4 Finite Wordlength Effects **12**-16
Number Representation • Fixed-Point Quantization Errors • Floating-Point Quantization Errors • Roundoff Noise • Limit Cycles • Overflow Oscillations • Coefficient Quantization Error • Realization Considerations

Jonathon A. Chambers
Sawasd Tantaratana
Bruce W. Bomar

12.1 Introduction

Digital filtering is concerned with the manipulation of discrete data sequences to remove noise, extract information, change the sample rate, and perform other functions. Although an infinite number of numerical manipulations can be applied to discrete data (e.g., finding the mean value, forming a histogram), the objective of digital filtering is to form a discrete output sequence $y(n)$ from a discrete input sequence $x(n)$. In some manner or another, each output sample is computed from the input sequence—not just from any one sample, but from many, in fact, possibly from all of the input samples. Those filters that compute their output from the present input and a finite number of past inputs are termed *finite impulse response* (FIR), whereas those that use all past inputs are *infinite impulse response* (IIR). This chapter will consider the design and realization of both FIR and IIR digital filters and will examine the effect of finite wordlength arithmetic on implementing these filters.

12.2 FIR Filters

A finite impulse response filter is a linear discrete time system that forms its output as the weighted sum of the most recent, and a finite number of past, inputs. Time-invariant FIR filters have finite memory, and their impulse response, namely, their response to a discrete input that is unity at the first sample and otherwise zero, matches the fixed weighting coefficients of the filter. Time-variant FIR filters, on the other hand, may operate at various sampling rates and/or have weighting coefficients that adapt in sympathy with some statistical property of the environment in which they are applied.

Fundamentals

Perhaps the simplest example of an FIR filter is the moving average operation described by the following linear constant-coefficient difference equation:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) \quad b_k = \frac{1}{M+1}$$

© 2006 by Taylor & Francis Group, LLC

where

$y(n)$ = output of the filter at integer sample index n
$x(n)$ = input to the filter at integer sample index n
b_k = filter weighting coefficients, $k = 0, 1, \ldots, M$
M = filter order

In a practical application, the input and output discrete-time signals is sampled at some regular sampling time interval, T seconds, denoted $x(nT)$ and $y(nT)$, which is related to the sampling frequency by $f_s = 1/T$ samples per second. For generality, however, it is more convenient to assume that T is unity, so that the effective sampling frequency is also unity and the Nyquist frequency (Oppenheim and Schafer, 1989), namely, the maximum analog frequency that when sampled at f_s will not yield an aliasing distortion, is one-half. It is then straightforward to scale, by multiplication, this normalized frequency range, that is, (0, 1/2), to any other sampling frequency.

The output of the simple moving average filter is the average of the $M + 1$ most recent values of $x(n)$. Intuitively, this corresponds to a smoothed version of the input, but its operation is more appropriately described by calculating the frequency response of the filter. First, however, the z-domain representation of the filter is introduced in analogy to the s- (or Laplace-) domain representation of analog filters. The z transform of a causal discrete-time signal $x(n)$ is defined by

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n}$$

where $X(z)$ is z transform of $x(n)$ and z is complex variable

The z transform of a delayed version of $x(n)$, namely, $x(n - k)$ with k a positive integer, is found to be given by $z^{-k}X(z)$. This result can be used to relate the z transform of the output $y(n)$ of the simple moving average filter to its input

$$Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z) \quad b_k = \frac{1}{M+1}$$

The z-domain transfer function, namely, the ratio of the output to input transform, becomes

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{M} b_k z^{-k} \quad b_k = \frac{1}{M+1}$$

Notice the transfer function $H(z)$ is entirely defined by the values of the weighting coefficients $b_k, k = 0, 1, \ldots, M$, which are identical to the discrete impulse response of the filter, and the complex variable z. The finite length of the discrete impulse response means that the transient response of the filter only lasts for $M + 1$ samples, after which steady state is reached. The frequency domain transfer function for the filter is found by setting $z = e^{j2\pi f}$, where $j = \sqrt{-1}$, and can be written as

$$H(e^{j2\pi f}) = \frac{1}{M+1} \sum_{k=0}^{M} e^{-j2\pi f k} = \frac{1}{M+1} e^{-j\pi f M} \frac{\sin[\pi f (M+1)]}{\sin(\pi f)}$$

The magnitude and phase response of the simple moving average filter, with $M = 7$, are calculated from $H(e^{j2\pi f})$ and shown in Fig. 12.1. The filter is seen clearly to act as a crude low-pass, smoothing filter with a **linear-phase** response. The sampling frequency periodicity in the magnitude and phase response is a property of discrete-time systems. The linear-phase response is due to the $e^{-j\pi f M}$ term in $H(e^{j2\pi f})$ and corresponds to a constant $M/2$ group delay through the filter. A phase discontinuity of $+/- 180^\circ$ is introduced each time the magnitude term changes sign. FIR filters that have center symmetry in their weighting coefficients have this constant, frequency independent group delay property that is very desirable in applications in which time dispersion is to be avoided, for example, in pulse transmission where it is important to preserve pulse shapes (Lee and Messerschmit, 1994).

© 2006 by Taylor & Francis Group, LLC

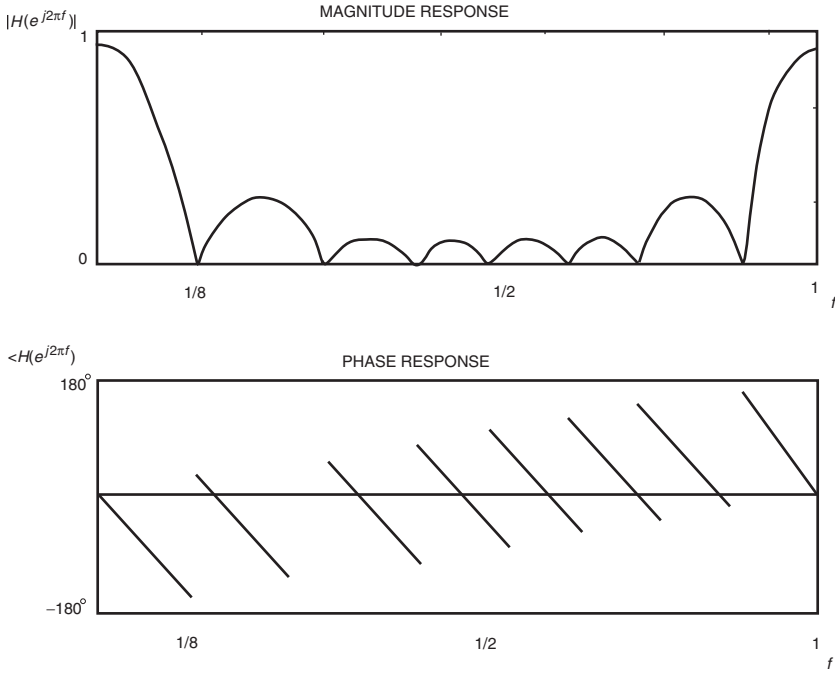


FIGURE 12.1 The magnitude and phase response of the simple moving average filter with $M = 7$.

Another useful interpretation of the z-domain transfer function is obtained by re-writing $H(z)$ as follows:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{M-k}}{z^M} = \frac{b_0 z^M + b_1 z^{M-1} + \cdots + b_{M-1} z + b_M}{z^M} = \frac{N(z)}{D(z)}$$

The z-domain transfer function is shown to be the ratio of two Mth-order polynomials in z, namely, $N(z)$ and $D(z)$. The values of z for which $N(z) = 0$ are termed the zeros of the filter, whereas those for which $D(z) = 0$ are the poles. The poles of such an FIR filter are at the origin, that is, $z = 0$, in the z plane. The positions of the zeros are determined by the weighting coefficients, that is, $b_k, k = 0, 1, \ldots, M$. The poles and zeros in the z plane for the simple moving average filter are shown in Fig. 12.2. The zeros, marked with a circle, are coincident with the unit circle, that is, the contour in the z plane for which $|z| = 1$, and match exactly the zeros in the magnitude response, hence their name; the discontinuities in the phase response are shown in Fig. 12.1. The zeros of an FIR filter may lie anywhere in the z plane because they do not impact on the stability of the filter; however, if the weighting coefficients are real and symmetric, or anti-symmetric, about their center value $M/2$, any complex zeros of the filter are constrained to lie as conjugate pairs coincident with the unit circle or as quartets of roots off the unit circle with the form $(\rho e^{j\theta}, \rho e^{-j\theta}, (1/\rho)e^{j\theta}, (1/\rho)e^{-j\theta})$ where ρ and θ are, respectively, the radius and angle of the first zero. Zeros that lie within the unit circle are termed *minimum*

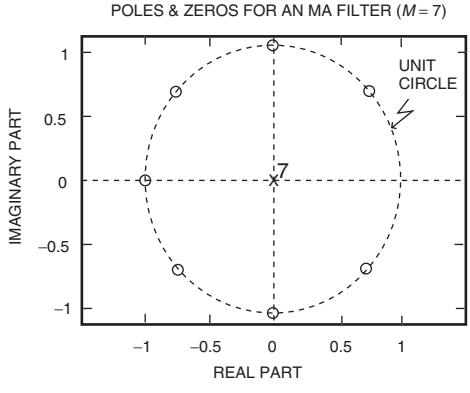


FIGURE 12.2 The pole-zero plot for the simple moving average filter with $M = 7$.

© 2006 by Taylor & Francis Group, LLC

phase, whereas those which lie outside the unit circle are called *maximum phase*. This distinction describes the contribution made by a particular zero to the overall phase response of a filter. A minimum-phase FIR filter that has all its zeros within the unit circle can have identical magnitude response to a maximum-phase FIR that has all its zeros outside the unit circle for the special case when they have an equal number of zeros with identical angles but reciprocal radii. An example of this would be second-order FIR filters with z-domain transfer functions $H_{\min}(z) = 1 + 0.5z^{-1} + 0.25z^{-2}$ and $H_{\max}(z) = 0.25 + 0.5z^{-1} + z^{-2}$. Notice that the center symmetry in the coefficients is lost, but the minimum- and maximum-phase weighting coefficients are simply reversed. Physically, a minimum-phase FIR filter corresponds to a system for which the energy is rapidly transferred from its input to its output, hence, the large initial weighting coefficients; whereas a maximum-phase FIR filter is slower to transfer the energy from its input to output; as such, its larger coefficients are delayed. Such FIR filters are commonly used for modeling multipath mobile communication transmission paths.

The characteristics of the frequency response of an FIR filter are entirely defined by the values of the weighting coefficients b_k, $k = 0, 1, \ldots, M$, which match the impulse response of the filter, and the order M. Various techniques are available for designing these coefficients to meet the specifications of some application. However, next consider the structures available for realizing an FIR filter.

Structures

The structure of an FIR filter must realize the z-domain transfer function given by

$$H(z) = \sum_{k=0}^{M} b_k z^{-k}$$

where z^{-1} is a unit delay operator. The building blocks for such filters are, therefore, adders, multipliers, and unit delay elements. Such elements do not have the disadvantages of analog components such as capacitors, inductors, operational amplifiers, and resistors, which vary with temperature and age. The direct or tapped-delay line form, as shown in Fig. 12.3, is the most straightforward realization of an FIR filter. The input $x(n)$ is delayed, scaled by the weighting coefficients b_k, $k = 0, 1, \ldots, M$, and accumulated to yield the output.

An equivalent, but transposed, structure of an FIR filter is shown in Fig. 12.4, which is much more modular and well suited to integrated circuit realization. Each module within the structure calculates a partial sum so that only a single addition is calculated at the output stage.

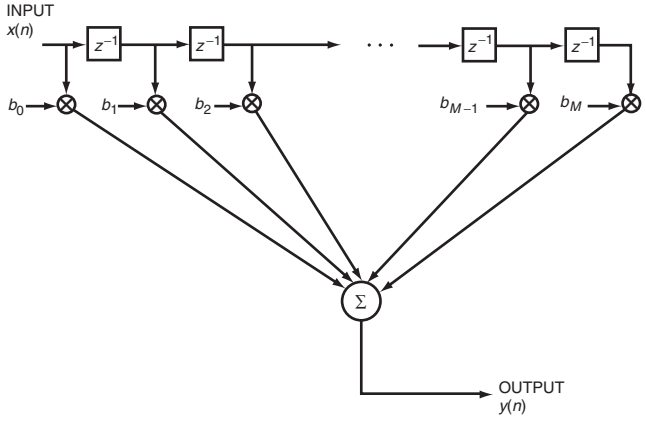


FIGURE 12.3 Direct form structure of an FIR filter.

© 2006 by Taylor & Francis Group, LLC

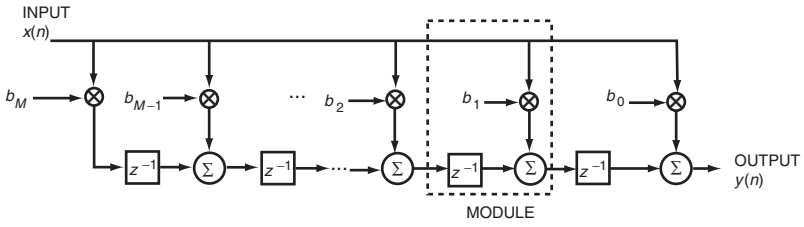


FIGURE 12.4 Modular structure of an FIR filter.

Other structures are possible that exploit symmetries or redundancies in the filter weighting coefficients. For example, FIR filters with linear phase have symmetry about their center coefficient $M/2$; approximately one-half of the coefficients of an odd length, that is, M even, half-band FIR filter, namely, a filter that ideally passes frequencies for $0 \leq f \leq 1/4$ and stops frequencies $1/4 < f \leq 1/2$, are zero. Finally, a lattice structure as shown in Fig. 12.5 may be used to realize an FIR filter. The multiplier coefficients within the lattice k_j, $j = 1, \ldots, M$, are not identical to the weighting coefficients of the other FIR filter structures but can be found by an iterative procedure. The attraction of the lattice structure is that it is staightforward to test whether all its zeros lie within the unit circle, namely, the minimum-phase property, and it has low sensitivity to quantization errors. These properties have motivated the use of lattice structures in speech coding applications (Rabiner and Schafer, 1978).

Design Techniques

Linear-phase FIR filters can be designed to meet various filter specifications, such as low-pass, high-pass, band-pass and bandstop filtering. For a low-pass filter, two frequencies are required, namely, the maximum frequency of the pass band below which the magnitude response of the filter is approximately unity, denoted the pass-band corner frequency f_p, and the minimum frequency of the stop band above which the magnitude response of the filter must be less than some prescribed level, named the stop-band corner frequency f_s. The difference between the pass- and stop-band corner frequencies is the *transition-bandwidth*. Generally, the order of FIR filter M required to meet some design specification increases with a reduction in the width of the transition band. There are three established techniques for coefficient design:

- Windowing
- Frequency sampling
- Optimal approximations

The windowing design method calculates the weighting coefficients by sampling the ideal impulse response of an analog filter and multiplying these values by a smoothing window to improve the overall

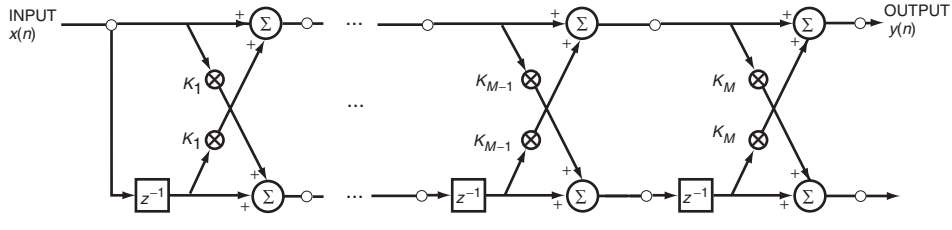


FIGURE 12.5 Lattice structure of an FIR filter.

© 2006 by Taylor & Francis Group, LLC

frequency domain response of the filter. The frequency sampling technique samples the ideal frequency domain specification of the filter and calculates the weighting coefficients by inverse transforming these values. However, better results can generally be obtained with the optimal approximationsmethod. The impulse response and magnitude response for a 40th-order optimal half-band FIR low-pass filter designed with the Parks-McClellan algorithm are shown in Fig. 12.6 together with the ideal frequency domain design specification. Notice the zeros in the impulse response. This algorithm minimizes the maximum deviation of the **magnitude response** of the design filter from the ideal magnitude response. The magnitude response of the design filter alternates about the desired specification within the pass band and above the specification in the stop band. The maximum deviation from the desired specification is equalized across the pass and stop bands; this is characteristic of an optimal solution.

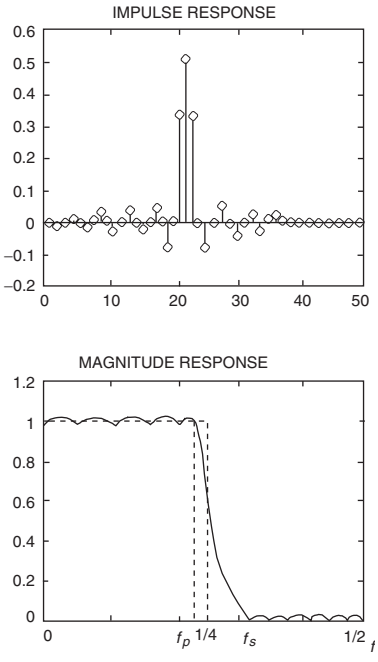


FIGURE 12.6 Impulse and magnitude response of an optimal 40th-order half-band FIR filter.

The optimal approximation approach can also be used to design *discrete-time differentiators* and *hilbert transformers*, namely, phase shifters. Such filters find extensive application in digital modulation schemes.

Multirate and Adaptive FIR Filters

FIR filter structures can be combined very usefully with multirate processing elements. Such a filtering scheme is, however, time variant and as such may introduce some distortion during processing, but with careful design this can be minimized. Consider the design of a low-pass filter with $f_c = 1/8$; frequencies above this value are essentially to be eliminated. The frequency range between f_c and the Nyquist frequency, that is, 1/2, therefore contains no useful information. The sampling frequency could, provided no aliasing distortion is introduced, therefore be usefully reduced by a factor of 4. Further processing, possibly adaptive, of the signal could then proceed at this reduced rate, hence, reducing the demands on the system design. This operation can be achieved in two ways: A time-invariant low-pass FIR filter, with $f_c = 1/8$, can be designed to operate at the original sampling rate, and its output can be down sampled, termed decimated, by a factor of four; or, more efficiently, the filtering can be performed in two stages, using the same simple half-band filter design twice, each operating at a lower sampling rate. These methods are shown diagrammatically in Fig. 12.7. The second scheme has considerable computational advantages and, because

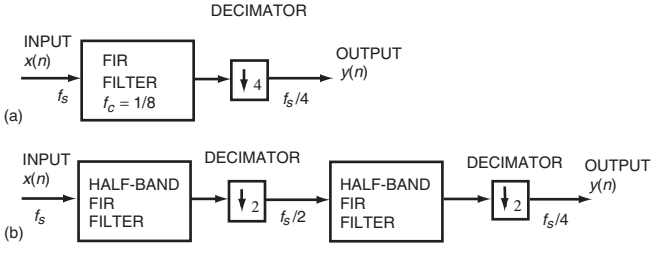


FIGURE 12.7 **Multirate FIR** low-pass **filter** structures: (a) method 1, (b) method 2.

© 2006 by Taylor & Francis Group, LLC

of the nature of half-band filters, it is also possible to move the decimators in front of the filters. With the introduction of modulation operations it is possible to use the same approach to achieve high-pass and band-pass filtering.

The basic structure of an **adaptive FIR filter** is shown in Fig. 12.8. The input to the adaptive filter $x(n)$ is used to make an estimate $\cap d(n)$ of the desired response $d(n)$. The difference between these quantities $e(n)$, is used by the adaptation algorithm to control the weighting coefficients of the FIR filter. The derivation of the desired response signal depends on the application; in channel equalization, as necessary for reliable communication with data modems, it is typically a training sequence known to the receiver. The weighting coefficients are adjusted to minimize some function of the error, such as the mean square value. The most commonly used adaptive algorithm is the least mean square (lms) algorithm that approximately minimizes the mean square error. Such filters have the ability to adapt to time-varying environments where fixed filters are inappropriate.

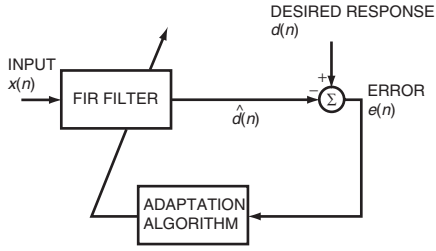


FIGURE 12.8 Adaptive FIR filter structure.

Applications

The absence of drift in the characteristics of digitally implemented FIR filters, their ability to reproduce, multirate realizations, and their ability to adapt to time-varying environments has meant that they have found many applications, particularly in telecommunications, for example, in receiver and transmitter design, speech compression and coding, and channel multiplexing. The primary advantages of fixed coefficient FIR filters are their unconditional stability due to the lack of feedback within their structure and their exact linear-phase characteristics. Nonetheless, for applications that require sharp, selective, filtering in standard form, they do require relatively large orders. For some applications, this may be prohibitive and, therefore, recursive; IIR filters are a valuable alternative.

12.3 Infinite Impulse Response (IIR) Filters

A digital filter with impulse response having infinite length is called an *infinite impulse response* (IIR) filter. An important class of IIR filters can be described by the difference equation

$$y(n) = b_0x(n) + b_1x(n-1) + \cdots + b_Mx(n-M) - a_1y(n-1) - a_2y(n-2) - \cdots - a_Ny(n-N) \quad (12.1)$$

where $x(n)$ is the input, $y(n)$ is the output of the filter, and $(a_1, a_2, \ldots, a_N)$ and $(b_0, b_1, \ldots, b_M)$ are real-value coefficients.

We denote the impulse response by $h(n)$, which is the output of the system when it is driven by a unit impulse at $n = 0$, with the system being initially at rest. The system function $H(z)$ is the z transform of $h(n)$. For the system in Eq. (12.1), it is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + \cdots + b_Mz^{-M}}{1 + a_1z^{-1} + a_2z^{-2} + \cdots + a_Nz^{-N}} \quad (12.2)$$

where N is called the filter order. Equation (12.28) can be put in the form of poles and zeros as

$$H(z) = b_0z^{N-M}\frac{(z-q_1)(z-q_2)\cdots(z-q_M)}{(z-p_1)(z-p_2)\cdots(z-p_N)} \quad (12.3)$$

The poles are at $p_1, p_2, \ldots, p_N$. The zeros are at $q_1, q_2, \ldots, q_M$, as well as $N - M$ zeros at the origin.

© 2006 by Taylor & Francis Group, LLC

The frequency response of the IIR filter is the value of the system function evaluated on the unit circle on the complex plane, that is, with $z = e^{j2\pi f}$, where f varies from 0 to 1, or from $-1/2$ to 1/2. The variable f represents the digital frequency. For simplicity, we write $H(f)$ for $H(z)|_{z=\exp(j2\pi f)}$. Therefore

$$H(f) = b_0 e^{j2\pi(N-M)f} \frac{(e^{j2\pi f} - q_1)(e^{j2\pi f} - q_2)\cdots(e^{j2\pi f} - q_M)}{(e^{j2\pi f} - p_1)(e^{j2\pi f} - p_2)\cdots(e^{j2\pi f} - p_N)} \tag{12.4}$$

$$= |H(f)|e^{j\theta(f)} \tag{12.5}$$

where $|H(f)|$ is the magnitude response and $\theta(f)$ is the phase response.

Compared to an FIR filter, an IIR filter requires a much lower order than an FIR filter to achieve the same requirement of the magnitude response. However, whereas an FIR filter is always stable, an IIR filter can be unstable if the coefficients are not properly chosen. Assuming that the system (12.1) is causal, then it is stable if all of the poles lie inside the unit circle on the z plane. Since the phase of a stable causal IIR filter cannot be made linear, FIR filters are chosen over IIR filters in applications where linear phase is essential.

Realizations

Equation (12.1) suggests a realization of an IIR filter as shown in Fig.12.9(a), which is called direct form I. By rearranging the structure, we can obtain direct form II, as shown in Fig. 12.9(b). Through transposition, we can obtain transposed direct form I and transposed direct form II, as shown in Fig 12.9(c) and Fig. 12.9(d).

The system function can be put in the form

$$H(z) = \prod_{i=0}^{K} \frac{b_{i0} + b_{i1}z^{-1} + b_{i2}z^{-2}}{1 + a_{i1}z^{-1} + a_{i2}z^{-2}} \tag{12.6}$$

by factoring the numerators and denominators into second-order factors, or in the form

$$H(z) = c_0 + \sum_{i=1}^{K} \frac{b_{i0} + b_{i1}z^{-1}}{1 + a_{i1}z^{-1} + a_{i2}z^{-2}} \tag{12.7}$$

by partial fraction expansion. The value of K is $N/2$ when N is even, and it is $(N+1)/2$ when N is odd. When N is odd, one of a_{i2} must be zero, as well as one of b_{i2} in Eq. (12.6) and one of b_{i1} in Eq. (12.7). According to Eq. (12.6), the IIR filter can be realized by K second-order IIR filters in cascade, as shown in Fig. 12.10(a). According to Eq. (12.7), the IIR filter can be realized by K second-order IIR filters and one scaler (i.e., c_0) in parallel, as depicted in Fig. 12.10(b). Each second-order subsystem can use any of the structures given in Fig. 12.9.

There are other realizations for IIR filters, such as state-space structure, wave structure, and lattice structure. See the references for details. In some situations, it is more convenient or suitable to use software realizations that are implemented by a digital signal processor.

IIR Filter Design

Designing an IIR filter involves choosing the coefficients to satisfy a given specification, usually a magnitude response specification. We assume that the specification is in the form depicted by Fig. 12.11, where the magnitude square must be in the range $(1/(1+\varepsilon^2), 1)$ in the pass band and it must be no larger than δ^2 in the stop band. The pass-band and the stop-band edges are denoted by f_p and f_s, respectively. No constraint is imposed on the response in the transition band, which lies between a pass band and a stop band.

There are various IIR filter design methods: design using an analog prototype filter, design using digital frequency transformation, and computer-aided design. In the first method, an analog filter is designed to meet the (analog) specification and the analog filter transfer function is transformed to digital system function. The second method assumes that some digital low-pass filter is available and the desired digital

© 2006 by Taylor & Francis Group, LLC

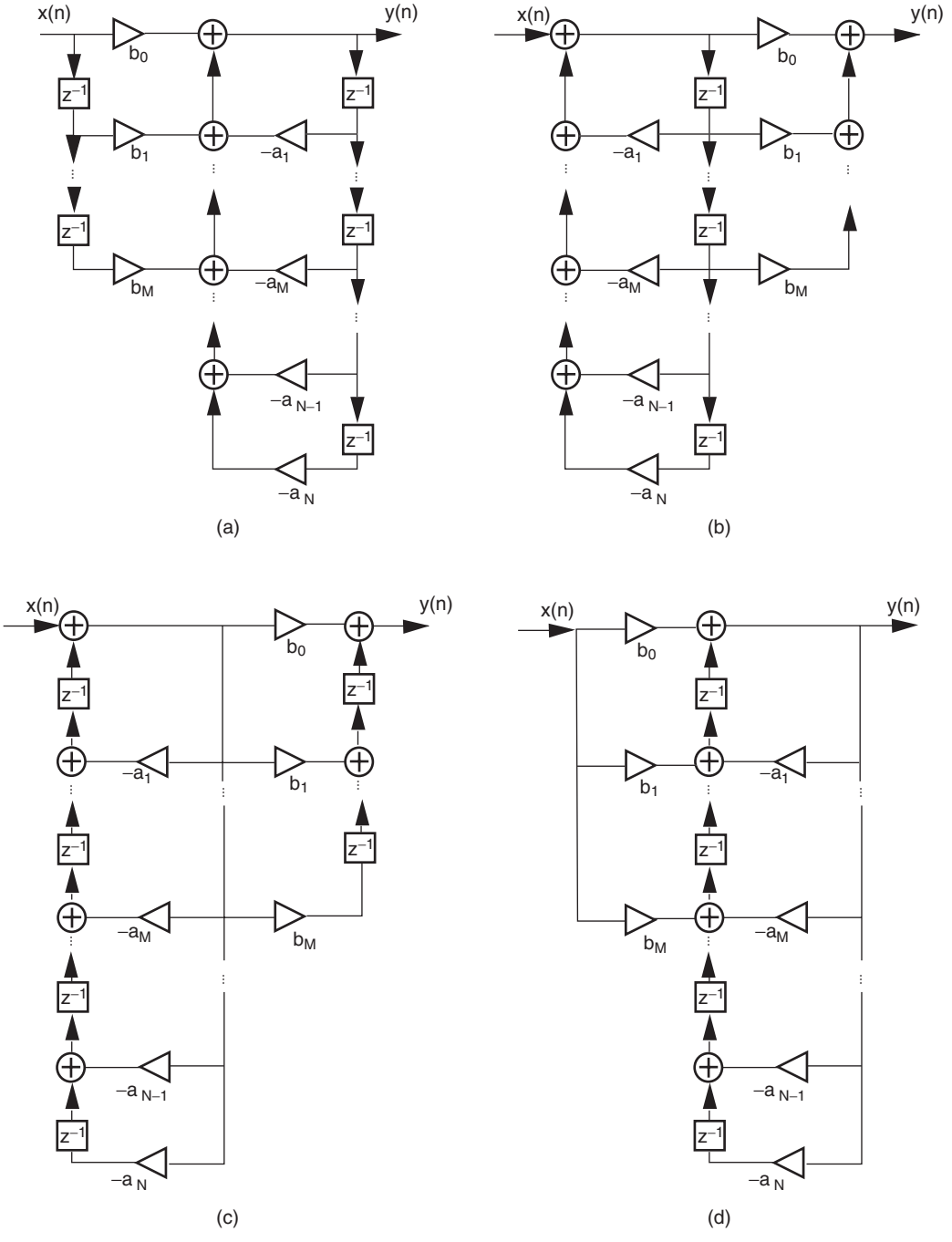


FIGURE 12.9 Direct form realizations of IIR filters: (a) direct form I, (b) direct form II, (c) transposed direct form I, (d) transposed direct form II.

filter is obtained from the digital low-pass filter by a digital frequency transformation. The last method involves algorithms that choose the coefficients so that the response is as close (in some sense) as possible to the desired filter. The first two methods are simple to do, and they are suitable for designing standard filters (low-pass, high-pass, bandpass, and bandstop filters). A computer-aided design requires computer

© 2006 by Taylor & Francis Group, LLC

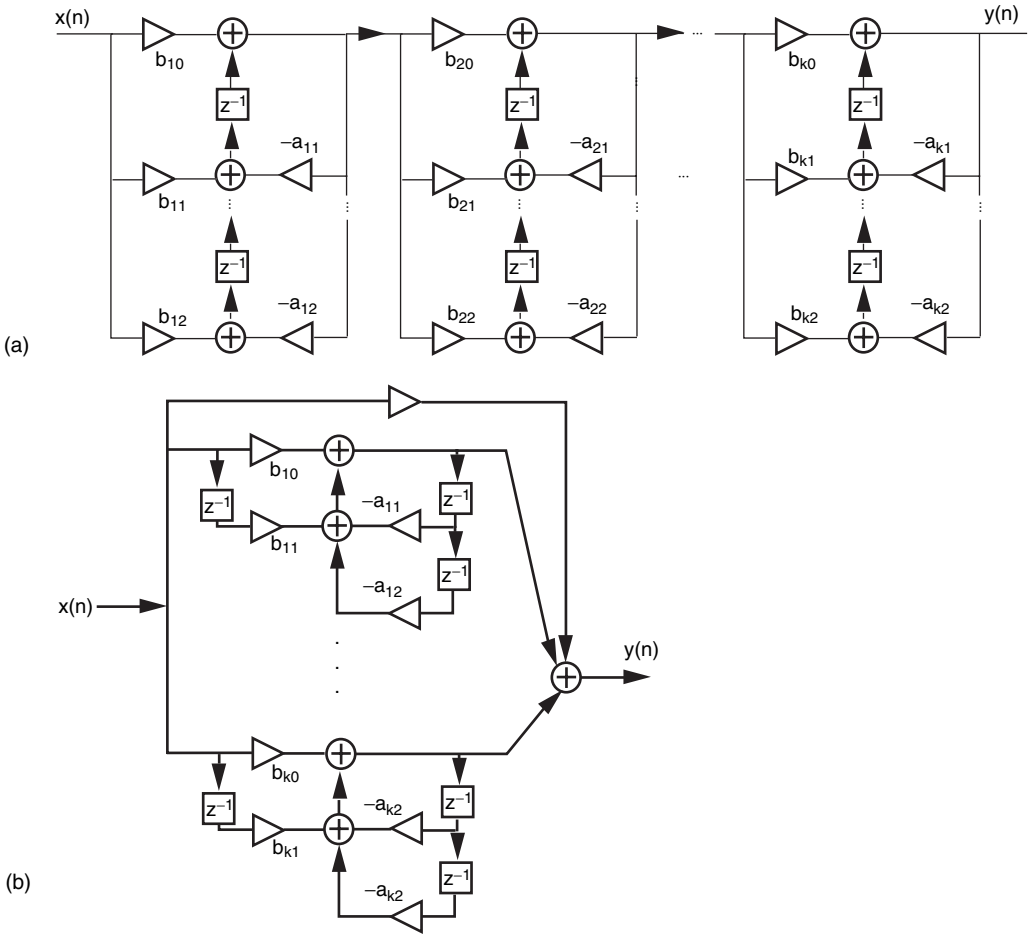


FIGURE 12.10 Realizations of IIR filters: (a) cascade form, (b) parallel form.

programming, but it can be used to design standard and nonstandard filters. We will focus only on the first method and present some design examples, as well as a summary of some analog filters, in the following sections.

Analog Filters

Here, we summarize three basic types of analog low-pass filters that can be used as prototype for designing IIR filters. For each type, we give the transfer function, its magnitude response, and the order N needed to satisfy the (analog) specification. We use $H_a(s)$ to denote the transfer function of an analog filter, where s is the complex variable in the Laplace transform. Each of these filters have all its poles on the left-half s plane, so that it is stable. We use the variable λ to represent the analog frequency in radians/second. The frequency response $H_a(\lambda)$ is the transfer function evaluated at $s = j\lambda$.

The analog low-pass filter specification, as shown in Fig. 12.12, is given by

$$\begin{aligned} (1+\varepsilon^2)^{-1} \le |H_a(\lambda)|^2 \le 1 \quad & \text{for } 0 \le (\lambda/2\pi) \le (\lambda_p/2\pi)\,\text{Hz} \\ 0 \le |H_a(\lambda)|^2 \le \delta^2 \quad & \text{for } (\lambda_s/2\pi) \le (\lambda/2\pi) < \infty\,\text{Hz} \end{aligned} \tag{12.8}$$

where λ_p and λ_s are the pass-band edge and stop-band edge, respectively.

© 2006 by Taylor & Francis Group, LLC

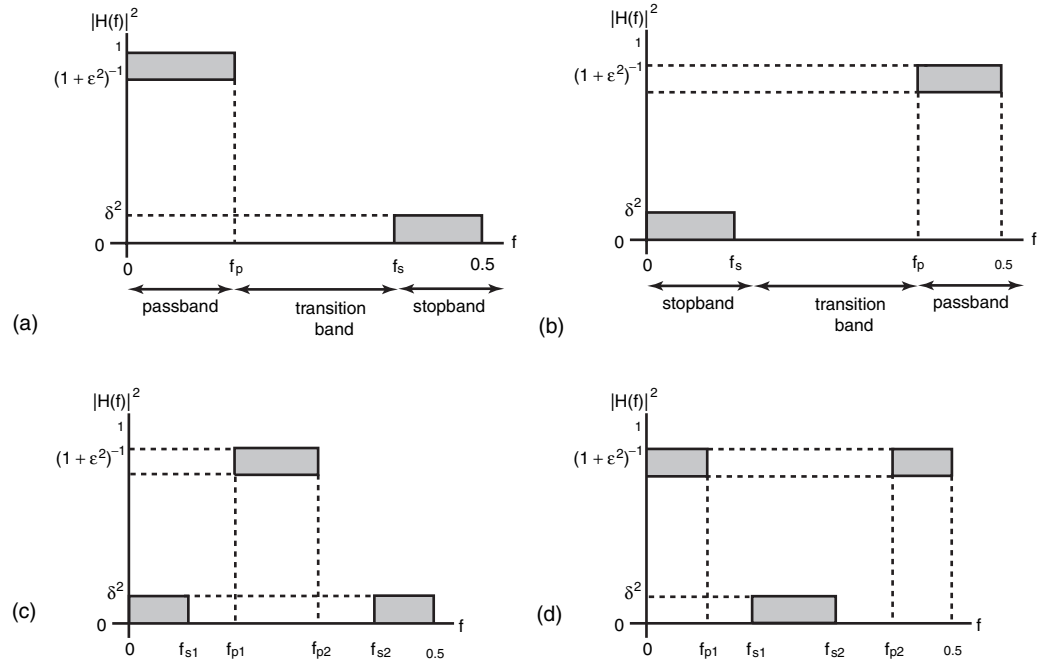


FIGURE 12.11 Specifications for digital IIR filters: (a) low-pass filter, (b) high-pass filter, (c) bandpass filter, (d) bandstop filter.

Butterworth Filters

The transfer function of an Nth-order Butterworth filter is given by

$$H_a(s) = \begin{cases} \displaystyle\prod_{i=1}^{N/2} \frac{1}{(s/\lambda_c)^2 - 2Re(s_i)(s/\lambda_c) + 1}, & N = \text{even} \\ \displaystyle\frac{1}{(s/\lambda_c) + 1} \prod_{i=1}^{(N-1)/2} \frac{1}{(s/\lambda_c)^2 - 2Re(s_i)(s/\lambda_c) + 1}, & N = \text{odd} \end{cases} \tag{12.9}$$

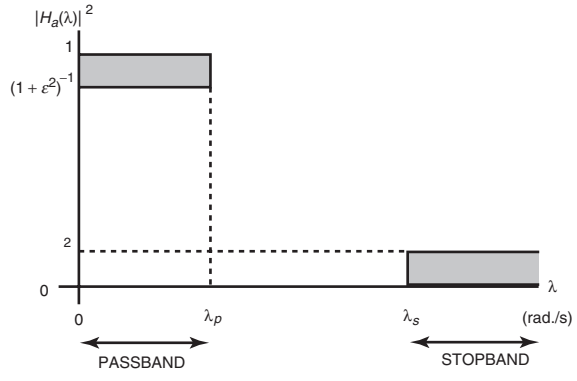


FIGURE 12.12 Specification for an analog low-pass filter.

© 2006 by Taylor & Francis Group, LLC

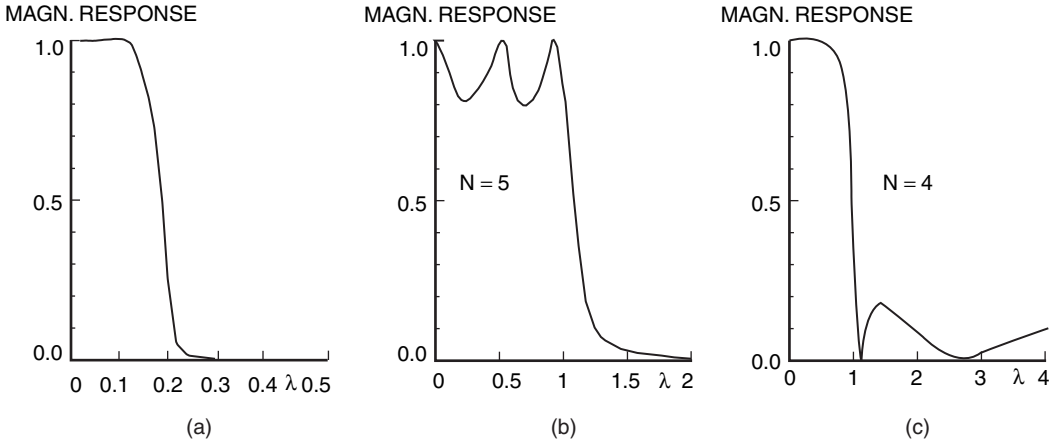


FIGURE 12.13 Magnitude responses of low-pass analog filters: (a) Butterworth filter, (b) Chebyshev filter, (c) inverse Chebyshev filter.

where $s_i = \exp\{j[1 + (2i - 1)/N]\pi/2\}$ and λ_c is the frequency where the magnitude drops by 3 dB. The magnitude response square is

$$|H_a(\lambda)|^2 = [1 + (\lambda/\lambda_c)^{2N}]^{-1} \tag{12.10}$$

Figure 12.13(a) shows the magnitude response $|H_a(\lambda)|$. To satisfy the specification in Eq. (12.8), the filter order is

$$N = \text{integer} \geq \frac{\log\left[\varepsilon/(\delta^{-2} - 1)^{\frac{1}{2}}\right]}{\log[\lambda_p/\lambda_s]} \tag{12.11}$$

and the value of λ_c is chosen from the following range:

$$\lambda_p \varepsilon^{-1/N} \leq \lambda_c \leq \lambda_s(\delta^{-2} - 1)^{-1/(2N)} \tag{12.12}$$

Chebyshev Filters (Type I Chebyshev Filters)

The Nth-order Chebyshev filter has a transfer function given by

$$H_a(s) = C \prod_{i=1}^{N} \frac{1}{(s - p_i)} \tag{12.13}$$

where

$$p_i = -\lambda_p \sinh(\phi)\sin\left(\frac{2i - 1}{2N}\pi\right) + j\lambda_p \cosh(\phi)\cos\left(\frac{2i - 1}{2N}\pi\right) \tag{12.14}$$

$$\phi = \frac{1}{N}\ln\left[\frac{1 + (1 + \varepsilon^2)^{\frac{1}{2}}}{\varepsilon}\right] \tag{12.15}$$

and where $C = -\prod_{i=1}^{N} p_i$ when N is odd and $(1+\varepsilon^2)^{-\frac{1}{2}} \prod_{i=1}^{N} p_i$ when N is even. Note that C normalizes the magnitude so that the maximum magnitude is 1. The magnitude square can be written as

$$|H_a(\lambda)|^2 = \left[1 + \varepsilon^2 T_N^2(\lambda/\lambda_p)\right]^{-1} \tag{12.16}$$

where $T_N(x)$ is the Nth-degree Chebyshev polynomial of the first kind that is given recursively by $T_0(x) = 1, T_1(x) = x$, and $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ for $n \geq 1$. Figure 12.13(b) shows an example of the magnitude response square. Notice that there are equiripples in the pass band. The filter

© 2006 by Taylor & Francis Group, LLC

order required to satisfy Eq. (12.8) is

$$N \geq \frac{\log\left\{\left[(\delta^{-2}-1)^{\frac{1}{2}}/\varepsilon\right] + \left[(\delta^{-2}-1)/\varepsilon^2 - 1\right]^{\frac{1}{2}}\right\}}{\log\{(\lambda_s/\lambda_p) + \left[(\lambda_s/\lambda_p)^2 - 1\right]^{\frac{1}{2}}} \tag{12.17}$$

which can be computed knowing $\varepsilon, \delta, \lambda_p$, and λ_s.

Inverse Chebyshev Filters (Type II Chebyshev Filters)

For inverse Chebyshev filters, the equiripples are inside the stop band, as opposed to the pass band in the case of Chebyshev filters. The magnitude response square of the inverse Chebyshev filter is

$$|H_a(\lambda)|^2 = \left[1 + (\delta^{-2}-1)/\,T_N^2(\lambda_s/\lambda)\right]^{-1} \tag{12.18}$$

Figure 12.13(c) depicts an example of Eq. (12.18). The value of $|H_a(\infty)|$ equals 0 if N is odd, and it equals δ if N is even. The transfer function giving rise to Eq. (12.18) is

$$H_a(s) = \begin{cases} C\displaystyle\prod_{i=1}^{N}\frac{(s-q_i)}{(s-p_i)}, & N \text{ is even} \\ \dfrac{C}{(s-p_{(N+1)/2})}\displaystyle\prod_{\substack{i=1 \\ i\neq(N+1)/2}}^{N}\frac{(s-q_i)}{(s-p_i)}, & N \text{ is odd} \end{cases} \tag{12.19}$$

where

$$p_i = \frac{\lambda_s}{\alpha_i^2+\beta_i^2}(\alpha_i - j\beta_i); \qquad q_i = j\frac{\lambda_s}{\cos\left(\frac{2i-1}{2N}\pi\right)} \tag{12.20}$$

$$\alpha_i = -\sinh(\phi)\sin\left(\frac{2i-1}{2N}\pi\right); \qquad \beta_i = \cosh(\phi)\cos\left(\frac{2i-1}{2N}\pi\right) \tag{12.21}$$

$$\phi = \frac{1}{N}\cosh^{-1}(\delta^{-1}) = \frac{1}{N}\ln\left[\delta^{-1} + (\delta^{-2}-1)^{\frac{1}{2}}\right] \tag{12.22}$$

and where

$$C = \prod_{i=1}^{N}(p_i/q_i)$$

when N is even and

$$C = -p_{(N+1)/2}\prod_{i=1, i\neq(N+1)/2}^{N}(p_i/q_i)$$

when N is odd. The filter order N required to satisfy Eq. (12.8) is the same as that for the Chebyshev filter, which is given by Eq. (12.17).

Comparison

In comparison, the Butterworth filter requires a higher order than both types of Chebyshev filters to satisfy the same specification. There is another type of filters called elliptic filters (Cauer filters) that have equiripples in the pass band as well as in the stop band. Because of the lengthy expressions, this type of filters is not given here (see the references). The Butterworth filter and the inverse Chebyshev filter have better (closer to linear) phase characteristics in the pass band than Chebyshev and elliptic filters. Elliptic filters require a smaller order than Chebyshev filters to satisfy the same specification.

© 2006 by Taylor & Francis Group, LLC

Design Using Bilinear Transformations

One of the most simple ways of designing a digital filter is by way of transforming an analog low-pass filter to the desired digital filter. Starting from the desired digital filter specification, the low-pass analog specification is obtained. Then, an analog low-pass filter $H_a(s)$ is designed to meet the specification. Finally, the desired digital filter is obtained by transforming $H_a(s)$ to $H(z)$. There are several types of transformation. The all-around best one is the bilinear transformation, which is the subject of this subsection.

In a bilinear transformation, the variable s in $H_a(s)$ is replaced with a bilinear function of z to obtain $H(z)$. Bilinear transformations for the four standard types of filters, namely, low-pass filter (LPF), high-pass filter (HPF), bandpass filter (BPF), and bandstop filter (BSF), are shown in Table 12.1. The second column in the table gives the relations between the variables s and z. The value of T can be chosen arbitrarily without affecting the resulting design. The third column shows the relations between the analog frequency λ and the digital frequency f, obtained from the relations between s and z by replacing s with $j\lambda$ and z with $\exp(j2\pi f)$. The fourth and fifth columns show the required pass-band and stop-band edges for the analog LPF. Note that the allowable variations in the pass band and stop band, or equivalently the values of ε and δ, for the analog low-pass filter remain the same as those for the desired digital filter. Notice that for the BPF and BSF, the transformation is performed in two steps: one for transforming an analog LPF to/from an analog BPF (or BSF), and the other for transforming an analog BPF (or BSF) to/from a digital BPF (or BSF). The values of W and $\tilde{\lambda}_0$ are chosen by the designer. Some convenient choices are: (1) $W = \tilde{\lambda}_{p2} - \tilde{\lambda}_{p1}$ and $\tilde{\lambda}_0^2 = \tilde{\lambda}_{p1}\tilde{\lambda}_{p2}$, which yield $\lambda_p = 1$; (2) $W = \tilde{\lambda}_{s2} - \tilde{\lambda}_{s1}$ and $\tilde{\lambda}_0^2 = \tilde{\lambda}_{s1}\tilde{\lambda}_{s2}$, which yield $\lambda_s = 1$. We demonstrate the design process by the following two examples.

TABLE 12.1 Bilinear Transformations for Standard Digital Filters

	Transformation (between s and z)	Frequency Relations (between f and λ)	Analog LPF Passband Edge	Analog LPF Stopband Edge
Digital LPF to/from analog LPF	$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$ $z = \frac{(2/T)+s}{(2/T)-s}$	$f = \frac{1}{\pi}\tan^{-1}\left(\frac{\lambda T}{2}\right)$ $\lambda = \frac{2}{T}\tan(\pi f)$	$\lambda_p = \frac{2}{T}\tan(\pi f_p)$	$\lambda_s = \frac{2}{T}\tan(\pi f_s)$
Digital HPF to/from analog LPF	$s = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}$ $z = \frac{s+(T/2)}{s-(T/2)}$	$f = \begin{cases} -\frac{1}{2} + \frac{1}{\pi}\tan^{-1}\left(\frac{2\lambda}{T}\right), & \lambda \geq 0 \\ \frac{1}{2} + \frac{1}{\pi}\tan^{-1}\left(\frac{2\lambda}{T}\right), & \lambda \leq 0 \end{cases}$ $\lambda = \frac{T}{2}\tan[\pi(f+0.5)], \lambda \geq 0$	$\lambda_p = \frac{T}{2}\tan\left[\pi\left(f_p + \frac{1}{2}\right)\right]$	$\lambda_s = \frac{T}{2}\tan\left[\pi\left(f_s + \frac{1}{2}\right)\right]$
Digital BPF to/from analog LPF	$s = \frac{\tilde{s}^2 + \tilde{\lambda}_0^2}{W\tilde{s}}$ where $\tilde{s} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$	$\lambda = \frac{\tilde{\lambda}^2 - \tilde{\lambda}_0^2}{W\tilde{\lambda}}$ where $\tilde{\lambda} = \frac{2}{T}\tan(\pi f)$	$\lambda_p = \max\left\{\left\lvert\frac{\tilde{\lambda}_0^2 - \tilde{\lambda}_{p1}^2}{W\tilde{\lambda}_{p1}}\right\rvert, \left\lvert\frac{\tilde{\lambda}_0^2 - \tilde{\lambda}_{p2}^2}{W\tilde{\lambda}_{p2}}\right\rvert\right\}$ where $\tilde{\lambda}_{p1} = \frac{2}{T}\tan(\pi f_{p1})$ $\tilde{\lambda}_{p2} = \frac{2}{T}\tan(\pi f_{p2})$	$\lambda_s = \min\left\{\left\lvert\frac{\tilde{\lambda}_0^2 - \tilde{\lambda}_{s1}^2}{W\tilde{\lambda}_{s1}}\right\rvert, \left\lvert\frac{\tilde{\lambda}_0^2 - \tilde{\lambda}_{s2}^2}{W\tilde{\lambda}_{s2}}\right\rvert\right\}$ where $\tilde{\lambda}_{s1} = \frac{2}{T}\tan(\pi f_{s1})$ $\tilde{\lambda}_{s2} = \frac{2}{T}\tan(\pi f_{s2})$
Digital BSF to/from analog LPF	$s = \frac{W\tilde{s}}{\tilde{s}^2 + \tilde{\lambda}_0^2}$ where $\tilde{s} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$	$\lambda = \frac{W\tilde{\lambda}}{\tilde{\lambda}_0^2 - \tilde{\lambda}^2}$ where $\tilde{\lambda} = \frac{2}{T}\tan(\pi f)$	$\lambda_p = \max\left\{\left\lvert\frac{W\tilde{\lambda}_{p1}}{\tilde{\lambda}_0^2 - \tilde{\lambda}_{p1}^2}\right\rvert, \left\lvert\frac{W\tilde{\lambda}_{p2}}{\tilde{\lambda}_0^2 - \tilde{\lambda}_{p2}^2}\right\rvert\right\}$ where $\tilde{\lambda}_{p1} = \frac{2}{T}\tan(\pi f_{p1})$ $\tilde{\lambda}_{p2} = \frac{2}{T}\tan(\pi f_{p2})$	$\lambda_s = \min\left\{\left\lvert\frac{W\tilde{\lambda}_{s1}}{\tilde{\lambda}_0^2 - \tilde{\lambda}_{s1}^2}\right\rvert, \left\lvert\frac{W\tilde{\lambda}_{s2}}{\tilde{\lambda}_0^2 - \tilde{\lambda}_{s2}^2}\right\rvert\right\}$ where $\tilde{\lambda}_{s1} = \frac{2}{T}\tan(\pi f_{s1})$ $\tilde{\lambda}_{s2} = \frac{2}{T}\tan(\pi f_{s2})$

© 2006 by Taylor & Francis Group, LLC

Design Example 1

Consider designing a digital LPF with a pass-band edge at $f_p = 0.15$ and stop-band edge at $f_s = 0.25$. The magnitude response in the pass band is to stay within 0 and -2 dB, although it must be no larger than -40 dB in the stop band. Assuming that the analog Butterworth filter is to be used as the prototype filter, we proceed the design as follows.

1. Compute $\varepsilon, \delta, \lambda_p, \lambda_s$, and the an alog filter order. An attenuation of -2d B means $10\log_{10}(1+\varepsilon^2)^{-1} = -2$, which yields $\varepsilon = 0.7648$, and attenuation of -40 dB means $10\log_{10}(\delta^2) = -40$, which gives $\delta = 0.01$. From Table 12.1, the analog pass-band and stop-band edges are $\lambda_p = (2/T)\tan(\pi f_p)$ and $\lambda_s = (2/T)\tan(\pi f_s)$. For convenience, we let $T = 2$, which makes $\lambda_p = 0.5095$ and $\lambda_s = 1.0$. Now, we can calculate the required order of the analog Butterworth filter from Eq. (12.11), which yields $N \geq 7.23$; thus, we choose $N = 8$.
2. Obtain the analog LPF transfer function. From Eq. (12.12), we can pick a value of λ_c between 0.5269 and 0.5623. Let us choose $\lambda_c = 0.54$. With $N = 8$, the transfer function is calculated from Eq. (12.9) as

$$H_a(s) = \frac{7.2302 \times 10^{-3}}{(s^2 + 0.2107s + 0.2916)(s^2 + 0.6s + 0.2916)} \bullet \frac{1}{(s^2 + 0.8980s + 0.2916)(s^2 + 1.0592s + 0.2916)} \tag{12.23}$$

3. Obtain the digital filter. Using the transformation in Table 12.8, we transform Eq. (12.23) to the desired digital filter by replacing s with $(2/T)(z-1)/(z+1) = (z-1)/(z+1)$ (since we let $T = 2$). After substitution and simplification, the resulting digital LPF has a system function given by

$$H(z) = H_a(s)|_{s=(z-1)(z+1)} = \frac{4.9428 \times 10^{-4}(z^2 + 2z + 1)^4}{(z^2 - 0.9431z + 0.7195)(z^2 - 0.7490z + 0.3656)} \bullet \frac{1}{(z^2 - 0.6471z + 0.1798)(z^2 - 0.6027z + 0.0988)} \tag{12.24}$$

   The magnitude response $|H(f)|$ is plotted in Fig. 12.14(a). The magnitude is 0.8467 at the pass-band edge $f_p = 0.15$, and it is 0.0072 at the stop-band edge $f_s = 0.25$, both of which are within the specification.

Design Example 2

Now, suppose that we wish to design a digital BSF with pass-band edges at $f_{p1} = 0.15$ and $f_{p2} = 0.30$ and stop-band edges at $f_{s1} = 0.20$ and $f_{s2} = 0.25$. The magnitude response in the pass bands is to stay within 0 and -2 dB, although it must be no larger than -40 dB in the stop band. Let us use the analog type I Chebyshev filter as the prototype filter. Following the same design process as the first example, we have the following.

1. Compute $\varepsilon, \delta, \lambda_p, \lambda_s$, and the analog filter order. From the first example, we have $\varepsilon = 0.7648$ and $\delta = 0.01$. From Table 12.1 and letting $T = 2$, we compute the analog pass-band and stop-band edges: $\tilde{\lambda}_{p1} = \tan(\pi f_{p1}) = 0.5095, \tilde{\lambda}_{p2} = \tan(\pi f_{p2}) = 1.3764, \tilde{\lambda}_{s1} = \tan(\pi f_{s1}) = 0.7265, \tilde{\lambda}_{s2} = \tan(\pi f_{s2}) = 1.0$. We choose $W = \tilde{\lambda}_{p2} - \tilde{\lambda}_{p1} = 0.8669$ and $\tilde{\lambda}_0^2 = \tilde{\lambda}_{p1}\tilde{\lambda}_{p2} = 0.7013$, so that $\lambda_p = 1.0$ and $\lambda_s = \min\{3.6311, 2.9021\} = 2.9021$. From Eq. (12.17), the required analog filter order is $N \geq 3.22$; thus, we choose $N = 4$.

© 2006 by Taylor & Francis Group, LLC

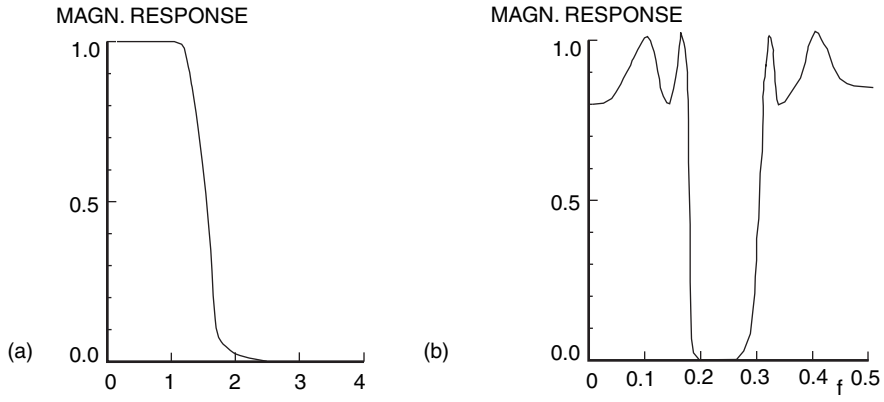


FIGURE 12.14 Magnitude responses of the designed digital filters in the examples: (a) LPF (example 1), (b) BSF (example 2).

2. Obtain the analog LPF transfer function. From Eq. (12.13) and with $N = 4$, we find the transfer function of the analog Chebyshev filter as

$$H_a(s) = \frac{1.6344 \times 10^{-1}}{(s^2 + 0.2098s + 0.9287)(s^2 + 0.5064s + 0.2216)} \tag{12.25}$$

3. Obtain the digital filter. Using Table 12.1, we transform Eq. (12.25) to $H(z)$ by replacing s with $W\tilde{s}/(\tilde{s}^2 + \tilde{\lambda}_0^2) = W(z^2 - 1)/[(z - 1)^2 + \tilde{\lambda}_0^2(z + 1)^2]$ since $T = 2$ and $\tilde{s} = (z - 1)/(z + 1)$. After substitution and simplification, the resulting digital BSF is

$$H(z) = \frac{1.7071 \times 10^{-1}(z^4 - 0.7023z^3 + 2.1233z^2 - 0.7023z + 1)^2}{(z^4 - 0.5325z^3 + 1.1216z^2 - 0.4746z + 0.8349)} \bullet \frac{1}{(z^4 - 0.3331z^3 + 0.0660z^2 - 0.0879z + 0.3019)} \tag{12.26}$$

The magnitude response is plotted in Fig. 12.14(b), which satisfies the specification.

12.4 Finite Wordlength Effects

Practical digital filters must be implemented with finite precision numbers and arithmetic. As a result, both the filter coefficients and the filter input and output signals are in discrete form. This leads to four types of finite wordlength effects.

Discretization (quantization) of the filter coefficients has the effect of perturbing the location of the filter poles and zeroes. As a result, the actual filter response differs slightly from the ideal response. This *deterministic* frequency response error is referred to as *coefficient quantization error*.

The use of finite precision arithmetic makes it necessary to quantize filter calculations by rounding or truncation. *Roundoff noise* is that error in the filter output that results from rounding or truncating calculations within the filter. As the name implies, this error looks like low-level noise at the filter output.

Quantization of the filter calculations also renders the filter slightly nonlinear. For large signals this nonlinearity is negligible and roundoff noise is the major concern. For recursive filters with a zero or constant input, however, this nonlinearity can cause spurious oscillations called *limit cycles*.

© 2006 by Taylor & Francis Group, LLC

With fixed-point arithmetic it is possible for filter calculations to overflow. The term *overflow oscillation*, sometimes also called *adder overflow limitcycle*, refers to a high-level oscillation that can existin another wise stable filter due to the nonlinearity associated with the overflow of internal filter calculations.

In this section we examine each of these finite wordlength effects for both fixed-point and floating-point number represention.

Number Representation

In digital signal processing, $(B+1)$-bit fixed-point numbers are usually represented as two's-complement signed fractions in the format

$$b_0.b_{-1}b_{-2}\cdots b_{-B}$$

The number represented is then

$$X = -b_0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-B}2^{-B} \tag{12.27}$$

where b_0 is the sign bit and the number range is $-1 \le X < 1$. The advantage of this representation is that the product of two numbers in the range from -1 to 1 is another number in the same range.

Floating-point numbers are represented as

$$X = (-1)^s m\, 2^c \tag{12.28}$$

where s is the sign bit, m is the *mantissa*, and c is the *characteristic* or *exponent*. To make the representation of a number unique, the mantissa is *normalized* so that $0.5 \le m < 1$.

Fixed-Point Quantization Errors

In fixed-point arithmetic, a multiply doubles the number of significant bits. For example, the product of the two 5-bit numbers 0.0011 and 0.1001 is the 10-bit number 00.00011011. The extra bit to the left of the decimal point can be discarded without introducing any error. However, the least significant four of the remaining bits must ultimately be discarded by some form of quantization so that the result can be stored to five bits for use in other calculations. In the preceding example this results in 0.0010 (quantization by rounding) or 0.0001 (quantization by truncating). When a sum of products calculation is performed, the quantization can be performed either after each multiply or after all products have been summed with double-length precision.

We will examine the case of fixed-point quantization by rounding. If X is an exact value then the rounded value will be denoted $Q_r(X)$. If the quantized value has B bits to the right of the decimal point, the quantization step size is

$$\Delta = 2^{-B} \tag{12.29}$$

Since rounding selects the quantized value nearest the unquantized value, it gives a value which is never more than $\pm\Delta/2$ away from the exact value. If we denote the rounding error by

$$\varepsilon_r = Q_r(X) - X \tag{12.30}$$

then

$$-\frac{\Delta}{2} \le \varepsilon_r \le \frac{\Delta}{2} \tag{12.31}$$

The error resulting from quantization can be modeled as a random variable uniformly distributed over the appropriate error range. Therefore, calculations with roundoff error can be considered

© 2006 by Taylor & Francis Group, LLC

error-free calculations that have been corrupted by additive white noise. The mean of this noise for rounding is

$$m_{\varepsilon_r} = E\{\varepsilon_r\} = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \varepsilon_r \, d\varepsilon_r = 0 \tag{12.32}$$

where $E\{\ \}$ represents the operation of taking the expected value of a random variable. Similarly, the variance of the noise for rounding is

$$\sigma_{\varepsilon_r}^2 = E\left\{(\varepsilon_r - m_{\varepsilon_r})^2\right\} = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} (\varepsilon_r - m_{\varepsilon_r})^2 \, d\varepsilon_r = \frac{\Delta^2}{12} \tag{12.33}$$

Floating-Point Quantization Errors

With floating-point arithmetic it is necessary to quantize after both multiplications and additions. The addition quantization arises because, prior to addition, the mantissa of the smaller number in the sum is shifted right until the exponent of both numbers is the same. In general, this gives a sum mantissa that is too long and so must be quantized.

We will assume that quantization in floating-point arithmetic is performed by rounding. Because of the exponent in floating-point arithmetic, it is the relative error that is important. The relative error is defined as

$$\varepsilon_r = \frac{Q_r(X) - X}{X} = \frac{\varepsilon_r}{X} \tag{12.34}$$

Since $X = (-1)^s \, m \, 2^c$, $Q_r(X) = (-1)^s \, Q_r(m) \, 2^c$ and

$$\varepsilon_r = \frac{Q_r(m) - m}{m} = \frac{\varepsilon}{m} \tag{12.35}$$

If the quantized mantissa has B bits to the right of the decimal point, $|\varepsilon| < \Delta/2$ where, as before, $\Delta = 2^{-B}$. Therefore, since $0.5 \le m < 1$,

$$|\varepsilon_r| < \Delta \tag{12.36}$$

If we assume that ε is uniformly distributed over the range from $-\Delta/2$ to $\Delta/2$ and m is uniformly distributed over 0.5–1,

$$\begin{aligned} m_{\varepsilon_r} &= E\left\{\frac{\varepsilon}{m}\right\} = 0 \\ \sigma_{\varepsilon_r}^2 &= E\left\{\left(\frac{\varepsilon}{m}\right)^2\right\} = \frac{2}{\Delta}\int_{\frac{1}{2}}^{1}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \frac{\varepsilon^2}{m^2} d\varepsilon \, dm = \frac{\Delta^2}{6} = (0.167)2^{-2B} \end{aligned} \tag{12.37}$$

From Eq. (12.34) we can represent a quantized floating-point value in terms of the unquantized value and the random variable ε_r using

$$Q_r(X) = X(1 + \varepsilon_r) \tag{12.38}$$

Therefore, the finite-precision product X_1X_2 and the sum $X_1 + X_2$ can be written

$$fl(X_1X_2) = X_1X_2(1 + \varepsilon_r) \tag{12.39}$$

© 2006 by Taylor & Francis Group, LLC

and

$$fl(X_1 + X_2) = (X_1 + X_2)(1 + \varepsilon_r) \tag{12.40}$$

where ε_r is zero mean with the variance of Eq. (12.37).

Roundoff Noise

To determine the roundoff noise at the output of a digital filter we will assume that the noise due to a quantization is stationary, white, and uncorrelated with the filter input, output, and internal variables. This assumption is good if the filter input changes from sample to sample in a sufficiently complex manner. It is not valid for zero or constant inputs for which the effects of rounding are analyzed from a limit cycle perspective.

To satisfy the assumption of a sufficiently complex input, roundoff noise in digital filters is often calculated for the case of a zero-mean white noise filter input signal $x(n)$ of variance σ_x^2. This simplifies calculation of the output roundoff noise because expected values of the form $E\{x(n)x(n-k)\}$ are zero for $k \neq 0$ and give σ_x^2 when $k = 0$. If there is more than one source of roundoff error in a filter, it is assumed that the errors are uncorrelated so that the output noise variance is simply the sum of the contributions from each source. This approach to analysis has been found to give estimates of the output roundoff noise that are close to the noise actually observed in practice.

Another assumption that will be made in calculating roundoff noise is that the product of two quantization errors is zero. To justify this assumption, consider the case of a 16-bit fixed-point processor. In this case a quantization error is of the order 2^{-15}, whereas the product of two quantization errors is of the order 2^{-30}, which is negligible by comparison.

The simplest case to analyze is a finite impulse response filter realized via the convolution summation

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{12.41}$$

When fixed-point arithmetic is used and quantization is performed after each multiply, the result of the N multiplies is N times the quantization noise of a single multiply. For example, rounding after each multiply gives, from Eqs. (12.29) and (12.33), an output noise variance of

$$\sigma_o^2 = N\frac{2^{-2B}}{12} \tag{12.42}$$

Virtually all digital signal processor integrated circuits contain one or more double-length accumulator registers that permit the sum-of-products in Eq. (12.41) to be accumulated without quantization. In this case only a single quantization is necessary following the summation and

$$\sigma_o^2 = \frac{2^{-2B}}{12} \tag{12.43}$$

For the floating-point roundoff noise case we will consider Eq. (12.41) for $N = 4$ and then generalize the result to other values of N. The finite-precision output can be written as the exact output plus an error term $e(n)$. Thus

$$\begin{aligned} y(n) + e(n) = (\{[h(0)x(n)(1 + \varepsilon_1(n)) + h(1)x(n-1)(1 + \varepsilon_2(n))][1 + \varepsilon_3(n)] \\ + h(2)x(n-2)(1 + \varepsilon_4(n))\}\{1 + \varepsilon_5(n)\} + h(3)x(n-3)(1 + \varepsilon_6(n)))(1 + \varepsilon_7(n)) \end{aligned} \tag{12.44}$$

In Eq. (12.44) $\varepsilon_1(n)$ represents the error in the first product, $\varepsilon_2(n)$ the error in the second product, $\varepsilon_3(n)$ the error in the first addition, etc. Notice that it has been assumed that the products are summed in the order implied by the summation of Eq. (12.41).

© 2006 by Taylor & Francis Group, LLC

Expanding Eq. (12.44), ignoring products of error terms, and recognizing $y(n)$ gives

$$e(n) = h(0)x(n)[\varepsilon_1(n) + \varepsilon_3(n) + \varepsilon_5(n) + \varepsilon_7(n)] + h(1)x(n-1)[\varepsilon_2(n) + \varepsilon_3(n) + \varepsilon_5(n) + \varepsilon_7(n)] + h(2)x(n-2)[\varepsilon_4(n) + \varepsilon_5(n) + \varepsilon_7(n)] + h(3)x(n-3)[\varepsilon_6(n) + \varepsilon_7(n)] \quad (12.45)$$

Assuming that the input is white noise of variance σ_x^2 so that $E\{x(n)x(n-k)\}$ is zero for $k \neq 0$, and assuming that the errors are uncorrelated,

$$E\{e^2(n)\} = [4h^2(0) + 4h^2(1) + 3h^2(2) + 2h^2(3)]\sigma_x^2\sigma_{\varepsilon_r}^2 \quad (12.46)$$

In general, for any N,

$$\sigma_o^2 = E\{e^2(n)\} = \left[Nh^2(0) + \sum_{k=1}^{N-1}(N+1-k)h^2(k)\right]\sigma_x^2\sigma_{\varepsilon_r}^2 \quad (12.47)$$

Notice that if the order of summation of the product terms in the convolution summation is changed, then the order in which the $h(k)$ appear in Eq. (12.47) changes. If the order is changed so that the $h(k)$ with smallest magnitude is first, followed by the next smallest, etc., then the roundoff noise variance is minimized. Performing the convolution summation in nonsequential order, however, greatly complicates data indexing and so may not be worth the reduction obtained in roundoff noise.

Analysis of roundoff noise for IIR filters proceeds in the same way as for FIR filters. The analysis is more complicated, however, because roundoff noise arising in the computation of internal filter variables (state variables) must be propogated through a transfer function from the point of the quantization to the filter output. This is not necessary for FIR filters realized via the convolution summation since all quantizations are in the output calculation. Another complication in the case of IIR filters realized with fixed-point arithmetic is the need to scale the computation of internal filter variables to avoid their overflow. Examples of roundoff noise analysis for IIR filters can be found in Weinstein and Oppenheim (1969) and Oppenheim and Weinstein (1972) where it is shown that differences in the filter realization structure can make a large difference in the output roundoff noise. In particular, it is shown that IIR filters realized via the parallel or cascade connection of first- and second-order subfilters are almost always superior in terms of roundoff noise to a high-order direct form (single difference equation) realization. It is also possible to choose realizations that are optimal or near optimal in a roundoff noise sense (Mullins and Roberts, 1976; Jackson, Lindgren, and Kim, 1979). These realizations generally require more computation to obtain an output sample from an input sample, however, and so suboptimal realizations with slightly higher roundoff noise are often preferable (Bomar, 1985).

Limit Cycles

A limit cycle, sometimes referred to as a *multiplier roundoff limit cycle*, is a low-level oscillation that can exist in an otherwise stable filter as a result of the nonlinearity associated with rounding (or truncating) internal filter calculations (Parker and Hess, 1971). Limit cycles require recursion to exist and do not occur in nonrecursive FIR filters.

As an example of a limit cycle, consider the second-order filter realized by

$$y(n) = Q_r\left\{\frac{7}{8}y(n-1) - \frac{5}{8}y(n-2) + x(n)\right\} \quad (12.48)$$

where $Q_r\{\ \}$ represents quantization by rounding. This is a stable filter with poles at $0.4375 \pm j0.6585$. Consider the implementation of this filter with four-bit (three bits and a sign bit) two's complement fixed-point arithmetic, zero initial conditions ($y(-1) = y(-2) = 0$), and an input sequence $x(n) = \frac{3}{8}\delta(n)$

© 2006 by Taylor & Francis Group, LLC

where $\delta(n)$ is the unit impulse or unit sample. The following sequence is obtained:

$$y(0) = Q_r\left\{\frac{3}{8}\right\} = \frac{3}{8} \qquad y(1) = Q_r\left\{\frac{21}{64}\right\} = \frac{3}{8} \qquad y(2) = Q_r\left\{\frac{3}{32}\right\} = \frac{1}{8}$$

$$y(3) = Q_r\left\{-\frac{1}{8}\right\} = -\frac{1}{8} \qquad y(4) = Q_r\left\{-\frac{3}{16}\right\} = -\frac{1}{8} \qquad y(5) = Q_r\left\{-\frac{1}{32}\right\} = 0$$

$$y(6) = Q_r\left\{\frac{5}{64}\right\} = \frac{1}{8} \qquad y(7) = Q_r\left\{\frac{7}{64}\right\} = \frac{1}{8} \qquad y(8) = Q_r\left\{\frac{1}{32}\right\} = 0$$

$$y(9) = Q_r\left\{-\frac{5}{64}\right\} = -\frac{1}{8} \qquad y(10) = Q_r\left\{-\frac{7}{64}\right\} = -\frac{1}{8} \qquad y(11) = Q_r\left\{-\frac{1}{32}\right\} = 0$$

$$y(12) = Q_r\left\{\frac{5}{64}\right\} = \frac{1}{8} \qquad \ldots$$

Notice that although the input is zero except for the first sample, the output oscillates with amplitude $\frac{1}{8}$ and period 6.

Limit cycles are primarily of concern in fixed-point recursive filters. As long as floating-point filters are realized as the parallel or cascade connection of first- and second-order subfilters, limit cycles will generally not be a problem since limit cycles are practically not observable in first- and second-order systems implemented with 32-bit floating-point arithmetic (Bauer, 1993). It has been shown that such systems must have an extremely small margin of stability for limit cycles to exist at anything other than underflow levels, which are at an amplitude of less than 10^{-38} (Bauer, 1993).

There are at least three ways of dealing with limit cycles when fixed-point arithmetic is used. One is to determine a bound on the maximum limit cycle a mplitude, expressed as an integral number of quantization steps. It is then possible to choose a wordlength that makes the limit cycle amplitude acceptably low. Alternately, limit cycles can be prevented by randomly rounding calculations up or down (Buttner, 1976). This approach, however, is complicated to implement. The third approach is to properly choose the filter realization structure and then quantize the filter calculations using magnitude truncation (Bomar, 1994). This approach has the disadvantage of slightly increasing roundoff noise.

Overflow Oscillations

With fixed-point arithmetic it is possible for filter calculations to overflow. This happens when two numbers of the same sign add to give a value having magnitude greater than one. Since numbers with magnitude greater than one are not representable, the result overflows. For example, the two's complement numbers 0.101 ($\frac{5}{8}$) and 0.100 ($\frac{4}{8}$) add to give 1.001, which is the two's complement representation of $-\frac{7}{8}$.

The overflow characteristic of two's complement arithmetic can be represented as $R\{\ \}$ where

$$R\{X\} = \begin{cases} X - 2, & X \geq 1 \\ X, & -1 \leq X < 1 \\ X + 2, & X < -1 \end{cases} \tag{12.49}$$

For the example just considered, $R\{\frac{9}{8}\} = -\frac{7}{8}$.

An overflow oscillation, sometimes also referred to as an adder overflow limit cycle, is a high-level oscillation that can exist in an otherwise stable fixed-point filter due to the gross nonlinearity associated with the overflow of internal filter calculations (Ebert, Mazo, and Taylor, 1969). Like limit cycles, overflow oscillations require recursion to exist and do not occur in nonrecursive FIR filters. Overflow oscillations also do not occur with floating-point arithmetic due to the virtual impossibility of overflow.

As an example of an overflow oscillation, once again consider the filter of Eq. (12.48) with four-bit fixed-point two's complement arithmetic and with the two's complement overflow characteristic

© 2006 by Taylor & Francis Group, LLC

of Eq. (12.49):

$$y(n) = Q_r\left\{R\left\{\frac{7}{8}y(n-1) - \frac{5}{8}y(n-2) + x(n)\right\}\right\} \tag{12.50}$$

In this case we apply the input

$$x(n) = \left\{-\frac{3}{4}, -\frac{5}{8}, 0, 0, 0, \ldots\right\} \tag{12.51}$$

giving the output sequence

$$y(0) = Q_r\left\{R\left\{-\frac{3}{4}\right\}\right\} = Q_r\left\{-\frac{3}{4}\right\} = -\frac{3}{4} \qquad y(1) = Q_r\left\{R\left\{-\frac{41}{32}\right\}\right\} = Q_r\left\{\frac{23}{32}\right\} = \frac{3}{4}$$

$$y(2) = Q_r\left\{R\left\{\frac{9}{8}\right\}\right\} = Q_r\left\{-\frac{7}{8}\right\} = -\frac{7}{8} \qquad y(3) = Q_r\left\{R\left\{-\frac{79}{64}\right\}\right\} = Q_r\left\{\frac{49}{64}\right\} = \frac{3}{4}$$

$$y(4) = Q_r\left\{R\left\{\frac{77}{64}\right\}\right\} = Q_r\left\{-\frac{51}{64}\right\} = -\frac{3}{4} \qquad y(5) = Q_r\left\{R\left\{-\frac{9}{8}\right\}\right\} = Q_r\left\{\frac{7}{8}\right\} = \frac{7}{8}$$

$$y(6) = Q_r\left\{R\left\{\frac{79}{64}\right\}\right\} = Q_r\left\{-\frac{49}{64}\right\} = -\frac{3}{4} \qquad y(7) = Q_r\left\{R\left\{-\frac{77}{64}\right\}\right\} = Q_r\left\{\frac{51}{64}\right\} = \frac{3}{4}$$

$$y(8) = Q_r\left\{R\left\{\frac{9}{8}\right\}\right\} = Q_r\left\{-\frac{7}{8}\right\} = -\frac{7}{8} \qquad \ldots$$

This is a large-scale oscillation with nearly full-scale amplitude.

There are several ways to prevent overflow oscillations in fixed-point filter realizations. The most obvious is to scale the filter calculations so as to render overflow impossible. However, this may unacceptably restrict the filter dynamic range. Another method is to force completed sums-of-products to saturate at ± 1, rather than overflowing (Ritzerfeld, 1989). It is important to saturate only the completed sum, since intermediate overflows in two's complement arithmetic do not affect the accuracy of the final result. Most fixed-point digital signal processors provide for automatic saturation of completed sums if their *saturation arithmetic* feature is enabled. Yet another way to avoid overflow oscillations is to use a filter structure for which any internal filter transient is guaranteed to decay to zero (Mills, Mullis, and Roberts, 1978). Such structures are desirable anyway, since they tend to have low roundoff noise and be insensitive to coefficient quantization (Barnes, 1979).

Coefficient Quantization Error

Each filter structure has its own finite, generally nonuniform grids of realizable pole and zero locations when the filter coefficients are quantized to a finite wordlength. In general the pole and zero locations desired in a filter do not correspond exactly to the realizable locations. The error in filter performance (usually measured in terms of a frequency response error) resulting from the placement of the poles and zeroes at the nonideal but realizable locations is referred to as coefficient quantization error.

Consider the second-order filter with complex-conjugate poles

$$\lambda = r\, e^{\pm j\theta} = \lambda_r \pm j\lambda_i = r\cos(\theta) \pm jr\sin(\theta) \tag{12.52}$$

and transfer function

$$H(z) = \frac{1}{1 - 2r\cos(\theta)z^{-1} + r^2 z^{-2}} \tag{12.53}$$

realized by the difference equation

$$y(n) = 2\lambda_r\, y(n-1) - r^2\, y(n-2) + x(n) \tag{12.54}$$

Quantizing the difference equation coefficients results in a nonuniform grid of realizable pole locations in the z plane. The nonuniform grid is defined by the intersection of vertical lines corresponding to

© 2006 by Taylor & Francis Group, LLC

quantization of $2\lambda_r$ and concentric circles corresponding to quantization of $-r^2$. The sparseness of realizable pole locations near $z = \pm 1$ results in a large coefficient quantization error for poles in this region. By contrast, quantizing coefficients of the normal realization (Barnes, 1979) corrresponds to quantizing λ_r and λ_i resulting in a uniform grid of realizable pole locations. In this case large coefficient quantization errors are avoided for all pole locations.

It is well established that filter structures with low roundoff noise tend to be robust to coefficient quantization, and vice versa (Jackson, 1976; Rao, 1986). For this reason, the normal (uniform grid) structure is also popular because of its low roundoff noise.

It is well known that in a high-order polynomial with clustered roots the root location is a very sensitive function of the polynomial coefficients. Therefore, filter poles and zeroes can be much more accurately controlled if higher order filters are realized by breaking them up into the parallel or cascade connection of first- and second-order subfilters. One exception to this rule is the case of linear-phase FIR filters in which the symmetry of the polynomial coefficients and the spacing of the filter zeros around the unit circle usually permits an acceptable direct realization using the convolution summation.

Realization Considerations

Linear-phase FIR digital filters can generally be implemented with acceptable coefficient quantization sensitivity using the direct convolution sum method. When implemented in this way on a digital signal processor, fixed-point arithmetic is not only acceptable but may actually be preferable to floating-point arithmetic. Virtually all fixed-point digital signal processors accumulate a sum of products in a double-length accumulator. Thismeans that only a single quantization is necessary to compute an output. Floating-point arithmetic, on the other hand, requires a quantization after every multiply and after every add in the convolution summation. With 32-bit floating-point arithmetic these quantizations introduce a small enough error to be insignificant for most applications.

When realizing IIR filters, either a parallel or cascade connection of first- and second-order subfilters is almost always preferable to a high-order direct form realization. With the availability of low-cost floating-point digital signal processors, like the Texas Instruments TMS320C32, it is highly recommended that floating-point arithmetic be used for IIR filters. Floating-point arithmetic simultaneously eliminates most concerns regarding scaling, limit cycles, and overflow oscillations. Regardless of the arithmetic employed, a low roundoff noise structure should be used for the second-order sections. The use of a low roundoff noise structure for the second-order sections also tends to give a realization with low coefficient quantization sensitivity. First-order sections are not as critical in determining the roundoff noise and coefficient sensitivity of a realization, and so can generally be implemented with a simple direct form structure.

Defining Terms

Adaptive FIR filter: A finite impulse response structure filter with adjustable coefficients. The adjustment is controlled by an adaptation algorithm such as the least mean square (lms) algorithm. They are used extensively in adaptive echo cancellers and equalizers in communication sytems.

Causality: The property of a system that implies that its output can not appear before its input is applied. This corresponds to an FIR filter with a zero discrete-time impulse response for negative time indices.

Discrete time impulse response: The output of an FIR filter when its input is unity at the first sample and otherwise zero.

Group delay: The group delay of an FIR filter is the negative derivative of the phase response of the filter and is, therefore, a function of the input frequency. At a particular frequency it equals the physical delay that a narrow-band signal will experience passing through the filter.

Linear phase: The phase response of an FIR filter is linearly related to frequency and, therefore, corresponds to constant group delay.

Linear, time invariant (LTI): A system is said to be LTI if superposition holds, that is, its output for an input that consists of the sum of two inputs is identical to the sum of the two outputs that result

© 2006 by Taylor & Francis Group, LLC

from the individual application of the inputs; the output is not dependent on the time at that the input is applied. This is the case for an FIR filter with fixed coefficients.

Magnitude response: The change of amplitude, in steady state, of a sinusoid passing through the FIR filter as a function of frequency.

Multirate FIR filter: An FIR filter in which the sampling rate is not constant.

Phase response: The phase change, in steady state, of a sinusoid passing through the FIR filter as a function of frequency.

References

Antoniou, A. 1993. *Digital Filters Analysis, Design, and Applications*, 2nd ed. McGraw-Hill, New York.

Barnes, C.W. 1979. Roundoff noise and overflow in normal digital filters. *IEEE Trans. Circuits Syst.* CAS-26(3):154–159.

Bauer, P.H. 1993. Limit cycle bounds for floating-point implementations of second-order recursive digital filters. *IEEE Trans. Circuits Syst.–II* 40(8):493–501.

Bomar, B.W. 1985. New second-order state-space structures for realizing low roundoff noise digital filters. *IEEE Trans. Acoust., Speech, Signal Processing* ASSP-33(1):106–110.

Bomar, B.W. 1994. Low-roundoff-noise limit-cycle-free implementation of recursive transfer functions on a fixed-point digital signal processor. *IEEE Trans. Industrial Electronics* 41(1):70–78.

Buttner, M. 1976. A novel approach to eliminate limit cycles in digital filters with a minimum increase in the quantization noise. In *Proceedings of the 1976 IEEE International Symposium on Circuits and Systems*, pp. 291–294. IEEE, NY.

Cappellini, V., Constantinides, A.G., and Emiliani, P. 1978. *Digital Filters and their Applications*. Academic Press, New York.

Ebert, P.M., Mazo, J.E., and Taylor, M.G. 1969. Overflow oscillations in digital filters. *Bell Syst. Tech. J.* 48(9): 2999–3020.

Gray, A.H. and Markel, J.D., 1973. Digital lattice and ladder filter synthesis. *IEEE Trans. Acoustics, Speech and Signal Processing* ASSP-21:491–500.

Herrmann, O. and Schuessler, W. 1970. Design of nonrecursive digital filters with minimum phase. *Electronics Letters* 6(11):329–330.

IEEE DSP Committee. 1979. *Programs for Digital Signal Processing*. IEEE Press, New York.

Jackson, L.B. 1976. Roundoff noise bounds derived from coefficient sensitivities for digital filters. *IEEE Trans. Circuits Syst.* CAS-23(8):481–485.

Jackson, L.B., Lindgren, A.G., and Kim, Y. 1979. Optimal synthesis of second-order state-space structures for digital filters. *IEEE Trans. Circuits Syst.* CAS-26(3):149–153.

Lee, E.A. and Messerschmitt, D.G. 1994. *Digital Communications*, 2nd ed. Kluwer, Norwell, MA.

Macchi, O. 1995. *Adaptive Processing: The Least Mean Squares Approach with Applications in Telecommunications*, Wiley, New York.

Mills, W.T., Mullis, C.T., and Roberts, R.A. 1978. Digital filter realizations without overflow oscillations. *IEEE Trans. Acoust., Speech, Signal Processing*. ASSP-26(4):334–338.

Mullis, C.T. and Roberts, R.A. 1976. Synthesis of minimum roundoff noise fixed-point digital filters. *IEEE Trans. Circuits Syst.* CAS-23(9):551–562.

Oppenheim, A.V. and Schafer, R.W. 1989. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Oppenheim, A.V. and Weinstein, C.J. 1972. Effects of finite register length in digital filtering and the fast fourier transform. *Proc. IEEE* 60(8):957–976.

Parker, S.R. and Hess, S.F. 1971. Limit-cycle oscillations in digital filters. *IEEE Trans. Circuit Theory* CT-18(11):687–697.

Parks, T.W. and Burrus, C.S. 1987. *Digital Filter Design*. Wiley, New York.

Parks, T.W. and McClellan, J.H. 1972a. Chebyshev approximations for non recursive digital filters with linear phase. *IEEE Trans. Circuit Theory* CT-19:189–194.

© 2006 by Taylor & Francis Group, LLC

Parks, T.W. and McClellan, J.H. 1972b. A program for the design of linear phase finite impulse response filters. *IEEE Trans. Audio Electroacoustics* AU-20(3):195–199.

Proakis, J.G. and Manolakis, D.G. 1992. *Digital Signal Processing Principles, Algorithms, and Applications*, 2nd ed. MacMillan, New York.

Rabiner, L.R. and Gold, B. 1975. *Theory and Application of Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Rabiner, L.R. and Schafer, R.W. 1978. *Digital Processing of Speech Signals.* Prentice-Hall, Englewood Cliffs, NJ.

Rabiner, L.R. and Schafer, R.W. 1974. On the behavior of minimax FIR digital Hilbert transformers. *Bell Sys. Tech. J.* 53(2):361–388.

Rao, D.B.V. 1986. Analysis of coefficient quantization errors in state-space digital filters. *IEEE Trans. Acoust., Speech, Signal Processing* ASSP-34(1):131–139.

Ritzerfeld, J.H.F. 1989. A condition for the overflow stability of second-order digital filters that is satisfied by all scaled state-space structures using saturation. *IEEE Trans. Circuits Syst.* CAS-36(8): 1049–1057.

Roberts, R.A. and Mullis, C.T. 1987. *Digital Signal Processing*. Addison-Wesley, Reading, MA.

Vaidyanathan, P.P. 1993. *Multirate Systems and Filter Banks*. Prentice-Hall, Englewood Cliffs, NJ.

Weinstein, C. and Oppenheim, A.V. 1969. A comparison of round off noise in floating-point and fixed-point digital filter realizations. *Proc. IEEE* 57(6):1181–1183.

Further Information

Additional information on the topic of digital filters is available from the following sources.

IEEE Transactionson Signal Processing, amonthly publication of the Institute of Electrical and Electronics Engineers, Inc, Corporate Office, 345 East 47 Street, NY.

IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing, a monthly publication of the Institute of Electrical and Electronics Engineers, Inc, Corporate Office, 345 East 47 Street, NY.

The Institute of Electrical and Electronics Engineers holds an annual conference at worldwide locations called the International Conference on Acoustics, Speech and Signal Processing, ICASSP, Corporate Office, 345 East 47 Street, NY.

IEE Transactions on Vision, Image and Signal Processing, a monthly publication of the Institute of Electrical Engineers, Head Office, Michael Faraday House, Six Hills Way, Stevenage, UK.

Signal Processing, a publication of the European Signal Processing Society, Switzerland, Elsevier Science B.V., Journals Dept., P.O. Box 211, 1000 AE Amsterdam, The Netherlands.

In addition, the following books are recommended.

Bellanger, M. 1984. *Digital Processing of Signals: Theory and Practice*. Wiley, New York.

Burrus, C.S. et al. 1994. *Computer-Based Exercises for Signal Processing Using MATLAB*. Prentice-Hall, Englewood Cliffs, NJ.

Jackson, L.B. 1986. *Digital Filters and Signal Processing*. Kluwer Academic, Norwell, MA, 1986.

Oppenheim, A.V. and Schafer, R.W. 1989. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Widrow, B. and Stearns, S. 1985. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

© 2006 by Taylor & Francis Group, LLC

13

Multichip Module Technology

13.1 Introduction **13**-1
13.2 Multichip Module Technology Definitions **13**-1
13.3 Design, Repair, and Test **13**-8
13.4 When to Use Multichip Modules **13**-10
13.5 Issues in the Design of Multichip Modules **13**-11

Paul D. Franzon

13.1 Introduction

Multichip module (**MCM**) technology allows bare integrated circuits and passive devices to be mounted together on a common interconnecting substrate. An example is shown in Fig. 13.1. In this photograph, eight chips are **wire-bonded** onto an MCM. MCM technology, however, is not just about packaging chips together; it can lead to new capabilities and unique performance and cost advantages. The purposes of this chapter are as follows: to explain the different multichip module technologies, to show how MCMs can be employed to improve the price/performance of systems, and to indicate some of the more important issues in MCM-system design.

13.2 Multichip Module Technology Definitions

In its broadest sense, a multichip module is an assembly in which more than one integrated circuit (IC) are bare mounted on a common substrate. This definition is often narrowed in such a way as to imply that the common substrate provides a higher wiring density than a conventional **printed circuit board** (**PCB**). The main physical components are shown in Fig. 13.2 and can be described as follows:

1. The substrate technology provides the interconnect between the chips (ICs or die) and any discrete circuit elements, such as resistors, capacitors, and inductors.
2. The chip connection technology provides the means whereby signals and power are transferred between the substrate and the chips.
3. The MCM package technology is the housing that surrounds the MCM and allows signals, and power and heat to be conducted to the outside world.

What cannot be shown in Fig. 13.2 are several other components that are important to the success of an MCM:

1. The test technology used to ensure correct function of the bare die, the MCM substrate, and the assembled MCM.
2. The repair technology used to replace failed die, if any are detected after assembly.
3. The design technology.

© 2006 by Taylor & Francis Group, LLC

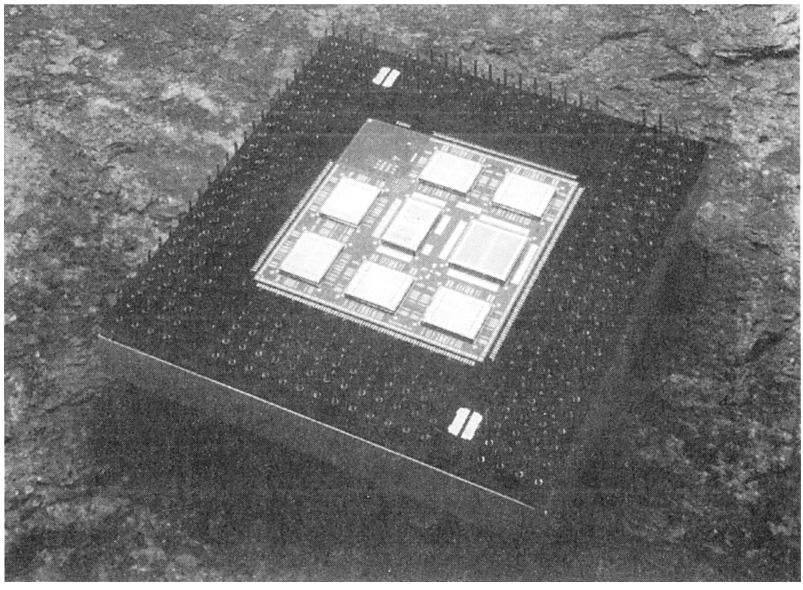


FIGURE 13.1 Eight chips wire-bonded into an MCM. (*Source:* MicroModule Systems.)

There are many different versions of these technology components. The substrate technology is broadly divided into three categories:

1. **Laminate MCMs (MCM-L)** is shown in (Fig. 13.3). Essentially fine-line PCBs, MCM-Ls are usually constructed by first patterning copper conductors on fiberglass/resin-impregnated sheets as shown in Fig. 13.4. These sheets are laminated together under heat and pressure. Connections between conductors on different sheets are made through via holes drilled in the sheets and plated. Recent developments in MCM-L technology have emphasized the use of flexible laminates. Flexible laminates have the potential for permitting finer lines and vias than fiberglass-based laminates.
2. **Ceramic MCMs (MCM-C)** are mostly based on pin grid array (PGA) technology. Typical cross sections and geometries are illustrated in Fig. 13.5 and the basic manufacturing steps are outline in Fig. 13.6. MCM-Cs are made by first casting a uniform sheet of prefired ceramic material, called green tape, then printing a metal ink onto the green tape, then punching and metal filling holes for the vias, and finally cofiring the stacked sheets together under pressure in an oven. In addition to metal, other inks can be used, including ones to print discrete resistors and capacitors in the MCM-C.
3. **Thin-film (deposited) MCMs (MCM-D)** are based on chip metalization processes. MCM-Ds have very fine feature sizes, giving high wiring densities (Fig. 13.7). MCM-Ds are made one layer at a time,

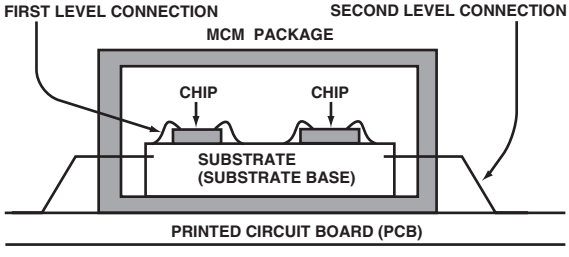


FIGURE 13.2 Physical technology components to an MCM.

© 2006 by Taylor & Francis Group, LLC

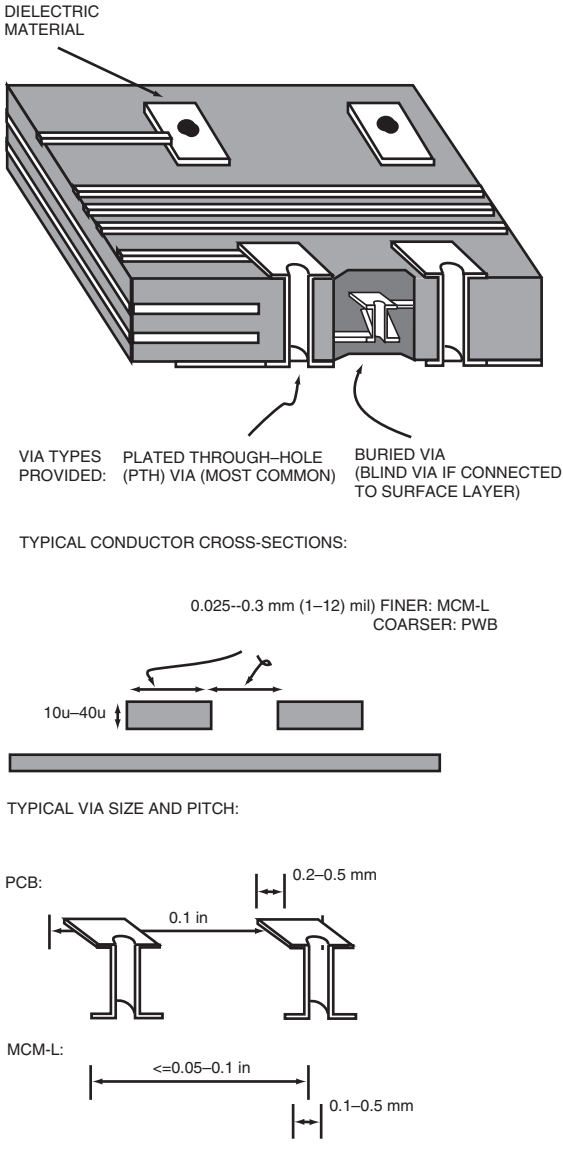


FIGURE 13.3 Typical cross sections and feature sizes in printed circuit board and MCM-L technologies.

using successive photolithographic definitions of metal conductor patterns and vias and the deposition of insulating layers, usually polyimide (Fig. 13.8). Often MCM-Ds are built on a silicon substrate, allowing capacitors, resistors, and transistors to be built, cheaply, as part of the substrate.

Table 13.1 gives one comparison of the alternative substrate technologies. MCM-Ls provide the lowest wiring and via density but are still very useful for making a small assembly when the total wire count in the design is not too high. MCM-Ds provide the highest wiring density and are useful in designs containing high pin-count chips; however, it is generally the most expensive technology on a per-unit-area basis. MCM-Cs fall somewhere in between, providing intermediate wiring densities at intermediate costs.

Current research promises to reduce the cost of MCM-Ls and MCM-D substrates. MCM-Ls based on flexible board technologies should be both cheaper and provide denser wiring than current fibre-glass MCM-L technologies. Though the nonrecurring costs of MCM-Ds will always be high, due to the

© 2006 by Taylor & Francis Group, LLC

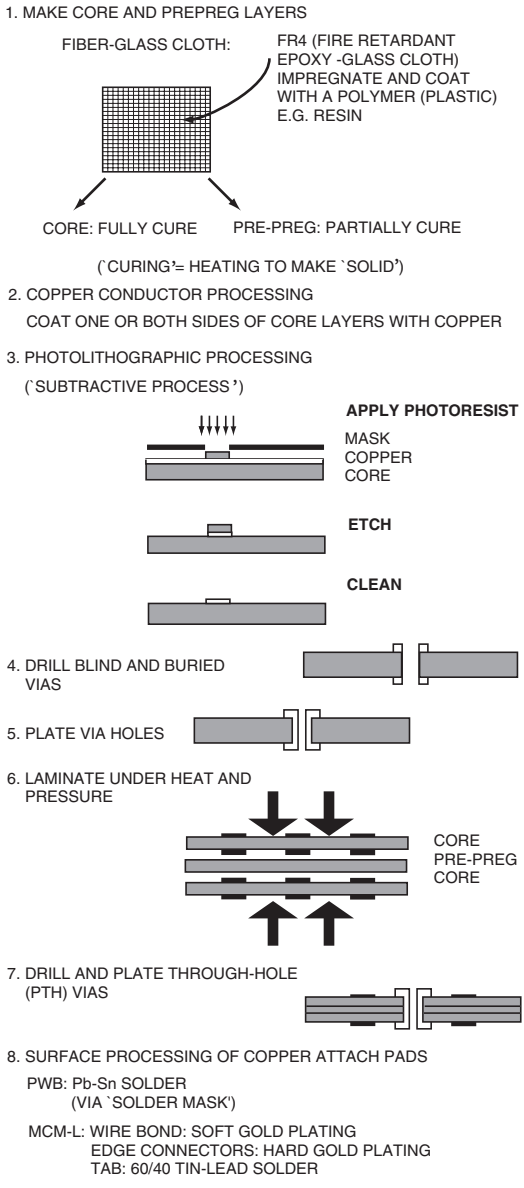


FIGURE 13.4 Basic manufacturing steps for MCM-Ls, using fiberglass/resin prepreg sheets as a basis.

requirement to make fine feature photolithographic masks, the cost per part is projected to decrease as more efficient manufacturing techniques are brought to fruition.

The chip to substrate connection alternatives are presented in Fig. 13.9. Currently, over 95% of the die mounted in MCMs or single chip packages are wire bonded. Most bare die come suitable for wire-bonding, and wire bonding techniques are well known. **Tape automated bonding (TAB)** is an alternative to wire bonding. With TAB assembly, the chip is first attached to the TAB frame inner leads. These leads are then shaped (called *forming*); then the outer leads can be attached to the MCM. TAB has a significant advantage over wire bonding in that the TAB-mounted chips can be more easily tested than bare die. The high tooling costs, however, for making the TAB frames, makes it less desirable in all but high volume production chips. The wire pitch in wire bonding or TAB is generally 75 μm or more.

© 2006 by Taylor & Francis Group, LLC

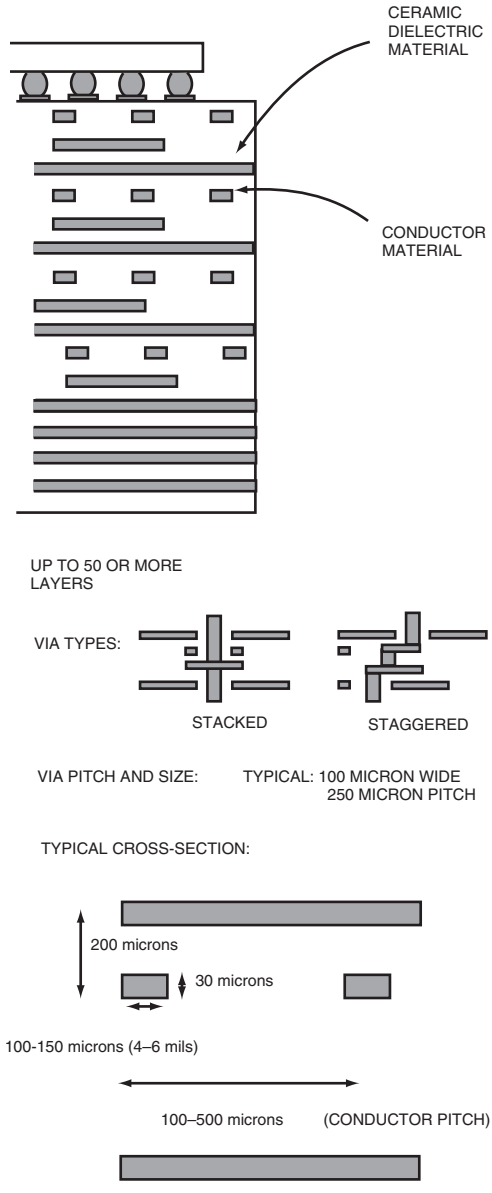


FIGURE 13.5 Typical cross sections and feature sizes in MCM-C technology.

With **flip-chip solder-bump** attachment, solder bumps are deposited over the area of the silicon wafer. The wafer is then diced into individual die and flipped over the MCM substrate. The module is then placed in a reflow oven, where the solder makes a strong connection between the chip and the substrate. Flip-chip attachment has the following advantages:

- Solder bumps can be placed over the entire area of the chip, allowing chips to have thousands of connections. For example, a 1-cm^2 chip could support 1600 solder bumps (at a conservative 250-μm pitch) but only 533 wire bonds.
- Transistors can be placed under solder bumps but cannot be placed under wire-bond or TAB pads. The reason arises from the relative attachment process. Making a good wire-bond or TAB lead

© 2006 by Taylor & Francis Group, LLC

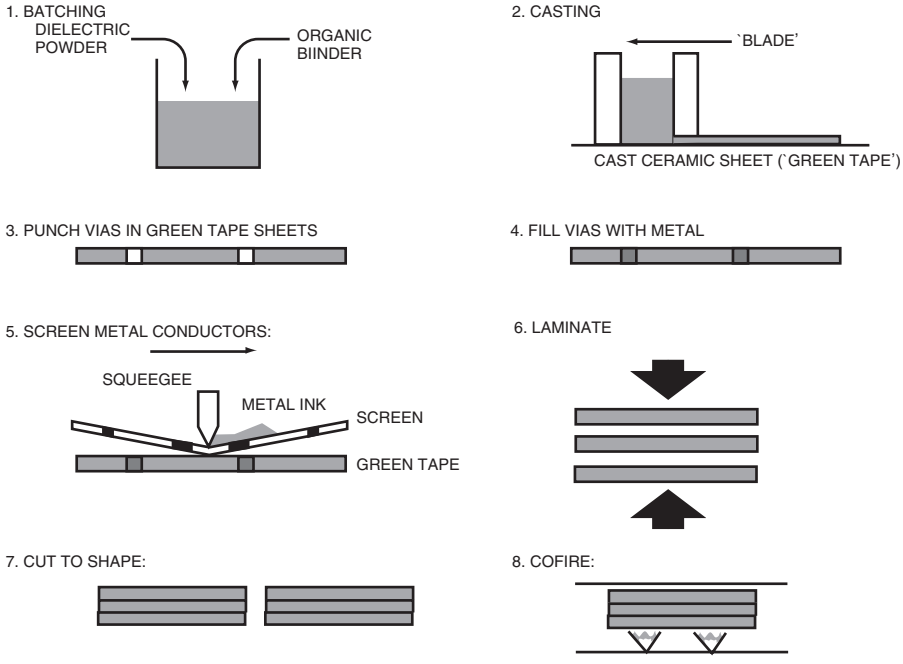


FIGURE 13.6 Basic manufacturing steps for MCM-Cs.

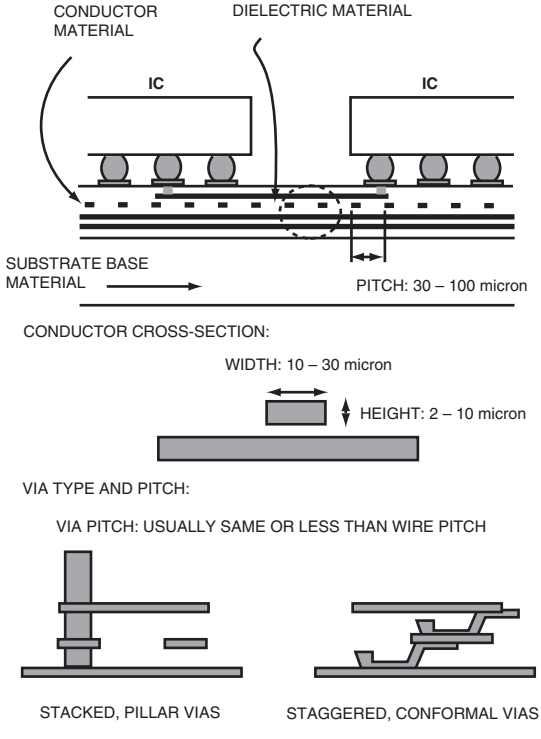


FIGURE 13.7 Typical cross sections and feature sizes in MCM-D technology.

© 2006 by Taylor & Francis Group, LLC

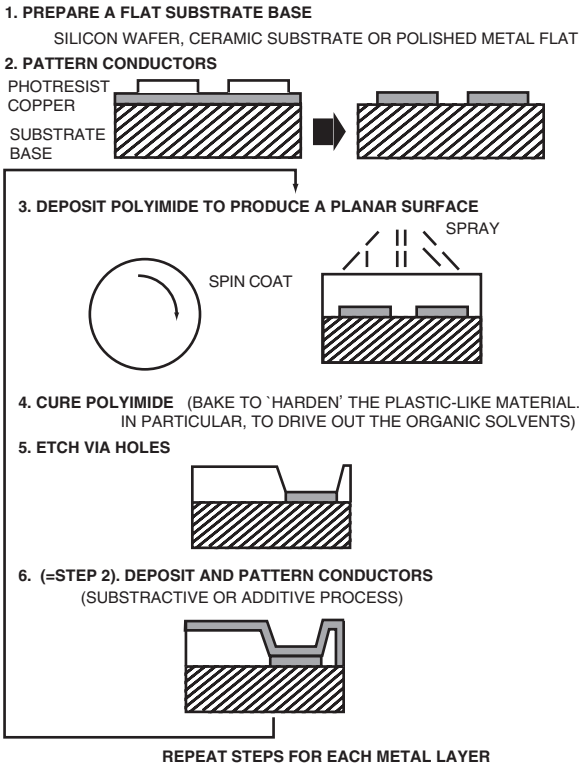

1. PREPARE A FLAT SUBSTRATE BASE
SILICON WAFER, CERAMIC SUBSTRATE OR POLISHED METAL FLAT
2. PATTERN CONDUCTORS
PHOTRESIST
COPPER
SUBSTRATE
BASE
3. DEPOSIT POLYIMIDE TO PRODUCE A PLANAR SURFACE
SPRAY
SPIN COAT
4. CURE POLYIMIDE (BAKE TO `HARDEN' THE PLASTIC-LIKE MATERIAL. IN PARTICULAR, TO DRIVE OUT THE ORGANIC SOLVENTS)
5. ETCH VIA HOLES
6. (=STEP 2). DEPOSIT AND PATTERN CONDUCTORS
(SUBSTRACTIVE OR ADDITIVE PROCESS)
REPEAT STEPS FOR EACH METAL LAYER

FIGURE 13.8 Basic manufacturing steps for MCM-Cs.

attachment requires compression of the bond into the chip pad, damaging any transistors beneath a pad. On the other hand, a good soldered connection is made through the application of heat only. As a result, the chip can be smaller by an area equivalent to the total pad ring area. For example, consider a 100-mm^2 chip with a 250-μm pad ring. The area consumed by the pad ring would total 10-mm^2, or 10% of the chip area. This area could be used for other functions if the chip were flipped and solder bumped. Alternatively, the chip could be made smaller and cheaper.

- The electrical parasitics of a solder bump are far better than for a wire-bond or TAB lead. The latter generally introduce about 1 nH of inductance and 1 pF of capacitance into a circuit. In contrast, a solder bump introduces about 10 pH and 10 nF. The lower parasitic inductance and capacitance make solder bumps attractive for high-frequency radio applications, for example, in the 5.6-GHz communications band and in high clock rate digital applications.
- The costs of flip-chip are comparable to wire bonding, beyound a certain point.

TABLE 13.1 Rough Comparison of the Three Different Substrate Technologies: 1 mil = 1/1000 in $\approx$ 25 μm

	Min. Wire Pitch, μm	Via Size, μm	Approx. Cost per Part \$/in^2	Nonrecurring Costs, \$
PCB	300	500	0.3	100
MCM-L	150	200	4	100–10,000
MCM-C	200	100	12	25,000
MCM-D	30	25	20	15,000

© 2006 by Taylor & Francis Group, LLC

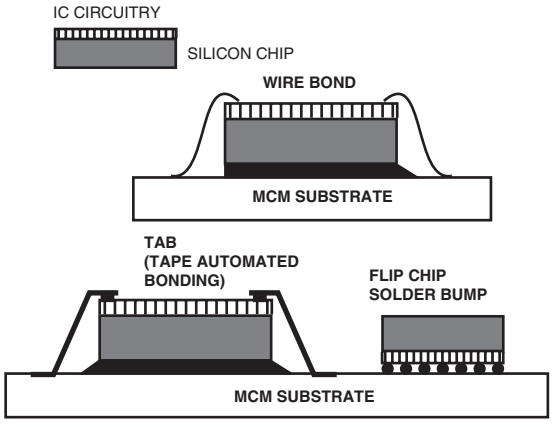


FIGURE 13.9 Chip attach options.

Flip-chip solder-bump, however, does have some disadvantages. The most significant follows from the requirement that the solder bump pads are larger than wire-bond pads (60–80 μm vs. approximately 50 μm). The solder-bump pads need to be larger in order to make the bumps taller. Taller bumps can more easily absorb the stresses that arise when the module is heated and cooled during assembly. As the thermal coefficient of expansion (TCE) of silicon is usually different from that of the MCM substrate, these two elements expand and contract at different rates, creating stresses in the connecting bumps. For the same reason, it is better to place the bumps in the center of the chip, rather than around the edge. By reducing the distance over which differential expansion and contraction occur, the stresses are reduced.

The larger bump pad sizes requires that the chips be designed specifically for solder-bumping, or that the wafers be postprocessed to distribute solder bumps pads over their surface, and wire (*redistribute*) these pads to the conventional wire-bond pads.

Another potential disadvantage arises from the lead content of the solder bump. Containing radioactive isotopes, most lead is a source of alpha particles that can potentially change the state of nearby transistors not shielded by aluminum. The effects of alpha particles are mainly of concern to dynamic random access memories (DRAMs) and dynamic logic.

13.3 Design, Repair, and Test

An important adjunct to the physical technology is the technology required to test and repair the die and modules. An important question that has to be answered for every MCM is how much to test the die before assembly vs. how much to rely on postassembly test to locate failed die. This is purely a cost question. The more the bare die are tested, the more likely that the assembled MCM will work. If the assembled MCM does not work, then it must either be repaired (by replacing a die) or discarded. The question reduces to the one that asks what level of bare die test provides a sufficiently high confidence that the assembled MCM will work, or is the assembled module cheap enough to throw away is a die is faulty.

In general, there are four levels of bare die test and burn-in, referred to as four levels of **known good die (KGD)**. In Table 13.2, these test levels are summarized, along with their impact. The lowest KGD level is to just use the same tests normally done at wafer level, referred to as the *wafer sort tests*. Here the chips are normally subject to a low-speed functional test combined with some parametric measurements (e.g., measurement of transistor curves). With this KGD level, test costs for bare die are limited to wafer test costs only. There is some risk, however, that the chip will not work when tested as part of the MCM. This risk is measured as the *test escape rate*. With conventional packaging, the chips are tested again, perhaps at full speed, once they are packaged, making the test escape rate zero.

© 2006 by Taylor & Francis Group, LLC

TABLE 13.2 Levels of Known Good Die and Their Impact

Known Good Die Level	Test Cost Impact	Test Escape Impact
Wafer level functional and parametric	Low	<1–2% fallout for mature ICs possibly >5% fallout for new ICs
At-pin speed sorted	Medium	Min. fallout for new digital ICs
Burned-in	High	Burn-in important for memories
Dynamically burned-in with full testing	Highest	Min. memory fallout

If the MCM contains chips that must meet tight timing specifications (e.g., a workstation) or the MCM must meet high reliability standards (e.g., for aerospace), however, then higher KGD test levels are required. For example, a workstation company will want the chips to be speed-sorted. Speed-sorting requires producing a test fixture that can carry high-speed signals to and from the tester. The test fixture type usually used at wafer sort, generally referred to as a *probe card*, is usually not capable of carrying high-speed signals. Instead, it is necessary to build a more expensive, high-speed test fixture, or temporarily mount the bare die in some temporary package for testing. Naturally, these additional expenses increase the cost of providing at-pin speed sorted die.

Some applications require even higher levels of known good die. Aerospace applications usually demand burn-in of the die, particularly for memories, so as to reduce the chance of infant mortality. There are two levels of burned-in KGD. In the lowest level, the die are stressed at high temperatures for a period of time and then tested. In the higher level, the die are continuously tested while in the oven.

How do you decide what test level is appropriate for your MCM? The answer is driven purely by cost; the cost of test vs. the cost of repair. For example, consider a 4-chip MCM using mature ICs. Mature ICs tend to have very high yields, and the process engineers have learned how to prevent most failure modes. As a result, the chances are very small that a mature IC would pass the waferlevel functional test and fail in the MCM. With a test escape rate of 2%, there is only a 8% ($1 - 0.98^4$) chance that each MCM would need to have a chip replaced after assembly. If the repair costs \$30 per replaced chip, then the average excess repair cost per MCM is \$2.40. It is unlikely that a higher level of KGD would add only \$0.60 to the cost of a chip. Thus, the lowest test level is justified.

On the other hand, consider an MCM containing four immature ICs. The functional and parametric wafer-level tests are poor at capturing speed faults. In addition, the process engineers have not had a chance to learn how to maximize the chip yield and how to best detect potential problems. If the test escape rate was 5%, there would be a 40% chance that a 4-chip MCM would require a chip to be replaced. The average repair cost would be \$12 per MCM in this scenario, and the added test cost of obtaining speed-sorted die would be justified.

For high-speed systems, speed sorting also greatly influences the speed rating of the module. Today, microprocessors, for example, are graded according to their clock rate. Faster parts command a higher price. In an MCM the entire module will be limited by the slowest chip on the module, and when a system is partitioned into several smaller chips, it is highly likely that a module will contain a slow chip if they are not tested.

For example, consider a set of chips that are manufactured in equal numbers of slow, medium, and fast parts. If the chips are speed sorted before assembly into the MCM, there should be 33% slow modules, 33% medium modules and 33% fast modules. If not speed sorted, these ratios change drastically as shown in Fig. 13.10. For a 4-chip module assembled from unsorted die, there will be 80% slow systems, 19% medium systems, and only 1% fast systems. This dramatic reduction in fast modules also justifies the need for speed-sorted die.

Design technology is also an important component to the MCM equation. Most MCM computer-aided design (CAD) tools have their basis in the PCB design tools. The main differences have been the inclusion of features to permit the use of the wide range of physical geometries possible in an MCM (particularly via geometries), as well as the ability to use bare die. A new format, the *die interchange format*, has been specifically developed to handle physical information concerning bare die (e.g., pad locations).

© 2006 by Taylor & Francis Group, LLC

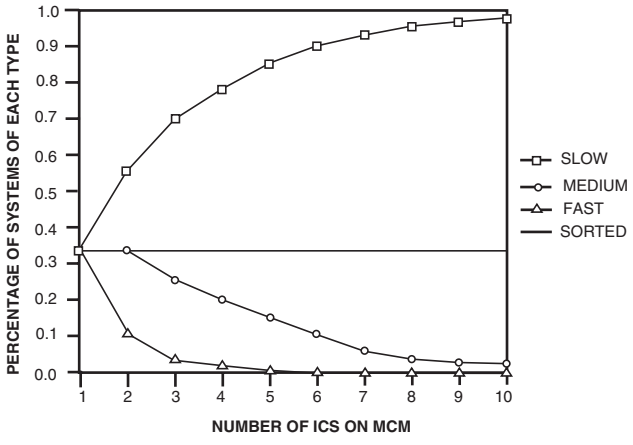


FIGURE 13.10 The effect of speed sorting on MCM performance.

There is more to MCM design technology, however, than just making small changes to the physical design tools (those tools that actually produce the MCM wiring patterns). Design correctness is more important in an MCM than in a PCB. For example, a jumper wire is difficult to place on an MCM in order to correct for an error. Thus, recent tool developments have concentrated on improving the designers ability to ensure that the multichip system is designed correctly before it is built. These developments include new simulation libraries that allow the designer to simulate the entire chip set before building the MCM, as well as tools that automate the electrical and thermal design of the MCM.

13.4 When to Use Multichip Modules

There are a number of scenarios given as follows that typically lead to consideration of an MCM alternative.

1. You must achieve a smaller form factor than is possible with single chip packaging. Often integrating the digital components onto an MCM-L provides the most cost effective way in which to achieve the required size. MCM-D provides the greatest form factor reduction when the utmost smallest size is needed.
2. The alternative is a single die that would either be too large to manufacture or would be so large as to have insufficient yield. The design might be custom or semicustom. In this case, partitioning the die into a number of smaller die and using an MCM to achieve the interconnect performance of the large die is an attractive alternative.
3. You have a technology mix that makes a single IC expensive or impossible and electrical performance is important. For example, you need to interface a complementary-metal-oxide-semiconductor (CMOS) digital IC with a GaAs microwave monolithic integrate circuit (MMIC) or high bandwidth analog IC. Or, you need a large amount of static random access memory (SRAM) and a small amount of logic. In these cases, an MCM might be very useful. An MCM-L might be the best choice if the number of interchip wires are small. A high layer count MCM-C or MCM-D might be the best choice if the required wiring density is large.
4. You are pad limited or speed limited between two ICs in the single chip version. For example, many computer designs benefit by having a very wide bus (256 bits or more) between the two levels of cache. In this case, an MCM allows a large number of very short connections between the ICs.
5. You are not sure that you can achieve the required electrical performance with single chip packaging. An example might be a bus operating above 100 MHz. If your simulations show that it will be difficult to guarantee the bus speed, then an MCM might be justified. Another example might be a mixed signal design in which noise control might be difficult. In general MCMs offer superior noise levels and interconnect speeds to single-chip package designs.

© 2006 by Taylor & Francis Group, LLC

6. The conventional design has a lot of solder joints and reliability is important to you. An MCM design has far fewer solder joints, resulting in less field failures and product returns. (Flip-chip solder bumps have shown themselves to be far more reliable than the solder joints used at board level.)

Though there are many other cases where the use of MCM technology makes sense, these are the main ones that have been encountered so far. If an MCM is justified, the next question might be to decide what ICs need to be placed on the MCM. A number of factors follow that need to be considered in this decision:

1. It is highly desirable that the final MCM package be compatible with single-chip packaging assembly techniques to facilitate manufacturability.
2. Although wires between chips on the MCM are inexpensive, off-MCM pins are expensive. The MCM should contain as much of the wiring as is feasible. On the other hand, an overly complex MCM with a high component count will have a poor final yield.
3. In a mixed signal design, separating the noisy digital components from the sensitive analog components is often desirable. This can be done by placing the digital components on an MCM. If an MCM-D with an integrated decoupling capacitor is used, then the on-MCM noise might be low enough to allow both analog and digital components to be easily mixed. Be aware in this case, however, that the on-MCM ground will have different noise characteristics than the PCB ground.

In short, most MCM system design issues are decided on by careful modeling of the system-wide cost and performance. Despite the higher cost of the MCM package itself, often cost savings achieved elsewhere in the system can be used to justify the use of an MCM.

13.5 Issues in the Design of Multichip Modules

The most important issue in the design of multichip modules is obtaining bare die. The first questions to be addressed in any MCM project are the questions as to whether the required bare die are available in the quantities required, with the appropriate test level, with second sourcing, at the right price, etc. Obtaining answers to these questions is time consuming as many chip manufacturers still see their bare die sales as satisfying a niche market. If the manufactured die are not properly available, then it is important to address alternative chip sources early.

The next most important issue is the test and verification plan. There are a number of contrasts with using a printed circuit board. First, as the nonrecurring engineering cost is higher for a multichip module, the desirability for first pass success is higher. Complete prefabrication design verification is more critical when MCMs are being used, so more effort must be spent on logic and electrical simulation prior to fabrication.

It is also important to determine, during design, how faults are going to be diagnosed in the assembled MCM. In a prototype, you wish to be able locate design errors before redoing the design. In a production module, you need to locate faulty die or wire bonds/solder bumps if a faulty module is to be repaired (typically by replacing a die). It is more difficult to physically probe lines on an MCM, however, than on a PCB. A fault isolation test plan must be developed and implemented. The test plan must be able to isolate a fault to a single chip or chip-to-chip interconnection. It is best to base such a plan on the use of chips with **boundary scan** (Fig. 13.11). With boundary scan chips, test vectors can be scanned in serially into registers around each chip. The MCM can then be run for one clock cycle, and the results scanned out. The results are used to determine which chip or interconnection has failed. If boundary scan is not available, and repair is viewed as necessary, then an alternative means for sensitizing between-chip faults is needed.

The decision as to whether a test is considered necessary is based purely on cost and test-escape considerations. Sandborn and Moreno (1994) and Ng in Donane and Franzon (1992, Chap. 4) provide more information. In general, if an MCM consists only of inexpensive, mature die, repair is unlikely to be worthwhile. The cost of repair is likely to be more than the cost of just throwing away the failed module. For MCMs with only one expensive, low-yielding die, the same is true, particularly if it is confirmed that the cause of most failures is that die. On the other hand, fault diagnosis and repair is usually desirable for modules containing multiple high-value die. You do not wish to throw all of these die away because only one failed.

© 2006 by Taylor & Francis Group, LLC

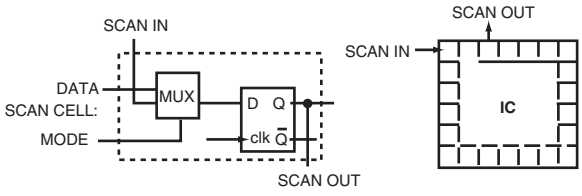


FIGURE 13.11 The use of boundary scan eases fault isolation in an MCM.

Thermal design is often important in an MCM. An MCM will have a higher heat density than the equivalent PCB, sometimes necessitating a more complex thermal solution. If the MCM is dissipating more than 1 W, it is necessary to check if any heat sinks and/or thermal spreaders are necessary.

Sometimes, this higher concentration of heat in an MCM can work to the designer's advantage. If the MCM uses one larger heat sink, as compared with the multiple heat sinks required on the single-chip packaged version, then there is the potential for cost savings.

Generally, electrical design is easier for a MCM than for a PCB; the nets connecting the chips are shorter, and the parasitic inductances and capacitances smaller. With MCM-D technology, it is possible to closely space the power and ground planes, so as to produce an excellent decoupling capacitor. Electrical design, however, cannot be ignored in an MCM; 300-MHz MCMs will have the save design complexity as 75-MHz PCBs.

MCMs are often used for mixed signal (mixed analog/RF and digital) designs. The electrical design issues are similar for mixed signal MCMs as for mixed signal PCBs, and there is a definite lack of tools to help the mixed signal designer. Current design practices tend to be qualitative. There is an important fact to remember that is unique to mixed signal MCM design. The on-MCM power and ground supplies are separated, by the package parasitics, from the on-PCB power and ground supplies. Many designers have assumed that the on-MCM and on-PCB references voltages are the same only to find noise problems appearing in their prototypes.

For more information on MCM design techniques, the reader is referred to Doane and Franzon (1992), Tummula and Rymaszewski (1989), and Messner et al. (1992).

Defining Terms

Boundary scan: Technique used to test chips after assembly onto a PCB or MCM. The chips are designed so that test vectors can be scanned into their input/output registers. These vectors are then used to determine if the chips, and the connections between the chips are working.

Ceramic MCM (MCM-C): A MCM built using ceramic packaging techniques.

Deposited MCM (MCM-D): A MCM built using the deposition and thin-film lithography techniques that are similar to those used in integrated circuit manufacturing.

Flip-chip solder-bump: Chip attachment technique in which pads and the surface of the chip have a solder ball placed on them. The chips are then flipped and mated with the MCM or PCB, and the soldered reflowed to create a solder joint. Allows area attachment.

Known good die (KGD): Bare silicon chips (die) tested to some known level.

Laminate MCM (MCM-L): A MCM built using advanced PCB manufacturing techniques.

Multichip module (MCM): A single package containing several chips.

Printed circuit board (PCB): Conventional board found in most electronic assemblies.

Tape automated bonding (TAB): A manufacturing technique in which leads are punched into a metal tape, chips are attached to the inside ends of the leads, and then the chip and lead frame are mounted on the MCM or PCB.

Wire bond: Where a chip is attached to an MCM or PCB by drawing a wire from each pad on the chip to the corresponding pad on the MCM or PCB.

© 2006 by Taylor & Francis Group, LLC

Acknowledgments

The author wishes to thank Andrew Stanaski for his helpful proofreading and for Fig. 13.11 and the comments related to it. He also wishes to thank Jan Vardaman and Daryl A. Doane for providing the tremendous learning experiences that lead to a lot of the knowledge provided here.

References

Dehkordi, P., Ramamurthi, K., Bouldin, D., Davidson, H., and Sandborn, P. 1995. Impact of packaging technology on system partitioning: A case study. In *1995 IEEE MCM Conference*, pp. 144–151.

Doane, D.A. and Franzon, P.D., eds. 1992. *Multichip Module Technologies and Alternatives: The Basics.* Van Nostrand Reinhold, New York.

Franzon, P.D., Stanaski, A., Tekmen, Y., and Banerjia. S. 1996. System design optimization for MCM. *Trans. CPMT.*

Messner, G., Turlik, I., Balde, J.W., and Garrou, P.E., eds. 1992. *Thin Film Multichip Modules.* ISHM.

Sandborn, P.A. and Moreno., H. 1994. *Conceptual Design of Multichip Modules and Systems.* Kluwer, Norwell, MA.

Tummula, R.R. and Rymaszewski, E.J., eds. 1989. *Microelectronics Packaging Handbook.* Van Nostrand Reinhold, Princeton, NJ.

Further Information

Additional information on the topic of multichip module technology is available from the following sources:

The major books in this area are listed in the References.

The primary journals are the *IEEE Transactions on Components, Packaging and Manufacturing Technology, Parts A and B,* and *Advancing Microelectronics*, published by ISHM.

The foremost trade magazine is *Advanced Packaging*, available free of charge to qualified subscribers.

The two main technical conferences are: (1) The IEEE Multichip Module Conference (MCMC), and (2) the ISHM/IEEE Multichip Module Conference.

© 2006 by Taylor & Francis Group, LLC

14

Testing of Integrated Circuits

14.1 Introduction **14**-1
14.2 Defect Types **14**-1
Traditional Faults and Fault Models
14.3 Concepts of Test **14**-3
Types of Test • Test Methods • External Stored Response Testing • BIST • Scan • I_{DDQ}
14.4 Test Tradeoffs **14**-8
Tradeoffs of Volume and Testability

Wayne Needham

14.1 Introduction

The goal of the test process for integrated circuits is to separate good devices from bad devices, and to test good devices as good. Bad devices tested as bad become yield loss, but they also represent opportunity for cost reduction as we reduce number of **defects** in the process. When good devices are tested as bad, overkill occurs and this directly impacts cost and profits. Finally, bad devices tested as good signal a quality problem, usually represented as *defects per million* (DPM) devices delivered.

Unfortunately, the test environment is not the same as the operating environment. The test process may reject good devices and may accept defective devices as good. Here lies one of the basic problems of today's test methods, and it must be understood and addressed as we look at how to generate and test integrated circuits. Figure 14.1 shows this miscorrelation of test to use.

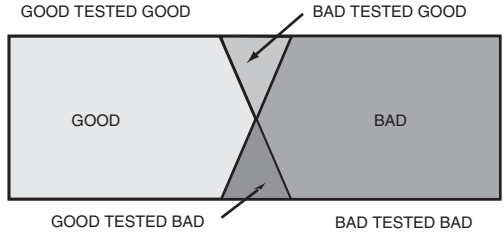


FIGURE 14.1 Test miscorrelations.

Bad devices are removed from good devices through a series of test techniques that enable the separation, by either voltage- or current-based testing. The match between the test process and the actual system use will not be addressed in this chapter.

14.2 Defect Types

Defects in an integrated circuit process fall into three broad categories. They are:

- Opens or abnormally high-resistance portions of signal lines and power lines that increase the series resistance between connected points. This extra resistance slows the charging of the parasitic

© 2006 by Taylor & Francis Group, LLC

TABLE 14.1 Typical IC Defects and Failure Modes

Defect	Source	Electrical Characteristic
Extra metal or interconnect	particle	ohmic connection between lines
Constrained geometry	insufficient or under etching	higher than normal resistance
Foreign material induced in the process in the process	contamination	varying resistance or short

capacitance of the line, thus resulting in a time delay. These defects also degrade voltage to a transistor circuit, which reduces levels or causes delays.

- Shorts, bridges, or resistive connections between layers of an integrated circuits, which form a resistance path in a normal insulating area. Given sufficiently low R value, the path may not be able to drive a 1 or a 0.
- Parameter variations such as threshold voltage and saturation current deviations changed by contamination, particles, implant, or process parameter changes. These usually result in speed or drive level changes.

Table 14.1 shows several types of defects and how they are introduced into an integrated circuit. It also describes electrical characteristics.

Defects, although common in integrated circuit processes, are difficult to quantify via test methods. Therefore, the industry has developed abstractions of defects called **faults**. Faults attempt to model the way defects manifest themselves in the test environment.

Traditional Faults and Fault Models

The most widely used fault model is the *single stuck-at fault* model. It is available in a variety of simulation environments with good support tools. The model represents a defect as a short of a signal line to a power supply line. This causes the signal to remain fixed at either a logic one or a zero. Thus, the logical result of an operation is not propagated to an output. Figure 14.2 shows the equivalent of a single stuck-at one or zero fault in a logic circuit. Although this fault model is the most common one, it was developed over 30 years ago, and does not always reflect today's more advanced processes. Today's very large-scale integrated (VLSI) circuits and submicron processes more commonly have failures that closely approximate bridges between signal lines. These bridges do not map well to the stuck-at fault model.

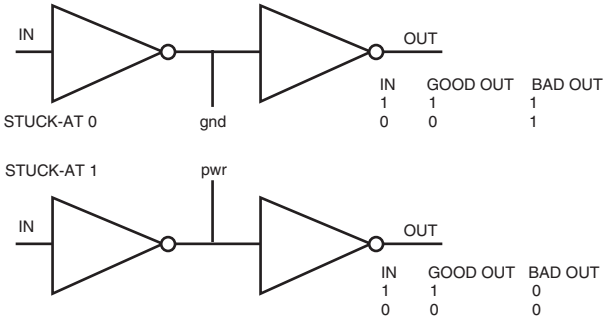


FIGURE 14.2 The single stuck-at model.

© 2006 by Taylor & Francis Group, LLC

Other proposed or available fault models include:

- The *open fault model:* This model is for open circuits. Unfortunately in CMOS, missing contacts and transistors can exhibit correct behavior when switching at speed, even if some of the transistors or contacts are missing.
- The *leakage fault model:* This model assumes leakage in transistors and lends itself to an I_{DDQ}[1] test technique (discussed later in this chapter).
- *Timing fault models* (delay, gate and transition faults): All of these models assume a change in the AC characteristics of the circuit. Either a single gate, line, or block of circuitry is slower than needed or simulated, and this slow signal causes a problem with at-speed operation of the integrated circuit.
- *Pseudostuck-at model fault model* for I_{DDQ}: This model assumes nodes look like they are stuck at a one or zero logic level.
- *Voting fault models*, and bridge models: Adjacent lines are assumed connected and the logic result is an X (unknown), or a vote with the highest drive strength circuit dominating the logic level.
- Also for memory circuits.
- *Neighborhood pattern faults*: This fault assumes operations in one memory cell will affect adjacent memory cells (usually limited to the four closest cells).
- *Coupling faults:* This fault model assumes that full rows or columns couple signals into the cells or adjacent rows and columns.
- *Retention faults:* These faults are similar to the open fault for logic (discussed previously). Failures to this fault model do not retain data over an extended period of time.

There are many other fault models.

It is important to realize that the stuck-at model may not necessarily catch failures and/or defects in the fabrication processes. This problem is being addressed by the computer-aided design (CAD) industry. Today, numerous solutions are available, or on the horizon, that address the issues of fault models. Tools currently available can support delay faults. Bridging fault routines have been demonstrated in university research, and should be available from selected CAD vendors within a few years. Soon, we will not be confined to testing solely through the single stuck-at model.

14.3 Concepts of Test

The main test method of an integrated circuit is to control and observe nodes. Doing this verifies the logical operation that ensures that the circuit is fault free. This process is usually done by generating stimulus patterns for the input and comparing the outputs to a known good state. Figure 14.3 demonstrates a classic case where a small portion of an integrated circuit is tested with a stimulus pattern set for inputs and an expected set for outputs. Note this test set is logic, not layout, dependent and checks only stuck at faults.

Types of Test

There are three basic types of test for integrated circuits. These are parametric tests, voltage tests, and current tests. Each has a place in the testing process.

Parametrics tests ensure the correct operation of the transistors (voltage and current) of the integrated circuit. Parametric tests include timing, input and output voltage, and current tests. Generally, they measure

[1] I_{DDQ} is the I_{IEEE} symbol for quotient current in a CMOS circuit.

© 2006 by Taylor & Francis Group, LLC

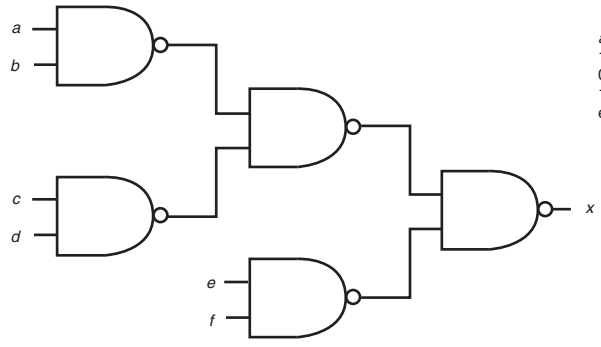


FIGURE 14.3 Logic diagram and test set.

circuit performance. In all of these tests, numbers are associated with the measurements. Typical readings include: $T_x < 334$ ns, $I_{OH} < 4$ mA, $V_{OI} < 0.4$ V, etc.

The next major category of testing is power supply current tests. Current tests include active switching current or power test, power down test, and standby or stable quiescent current tests. This last test, often called I_{DDQ}, will be discussed later.

The most common method of testing integrated circuits is the voltage-based test. Voltage-based test for logic operations includes functions such as placing a one on the input of an integrated circuit and ensuring that the proper one or zero appears on the output of the IC at the right time. Logical values are repeated, in most cases, at high and low operating voltage. The example in Fig. 14.3 is a voltage-based test.

Test Methods

To implement voltage based testing or any of the other types of tests, initialization of the circuit must take place, and control and observation of the internal nodes must occur. The following are four basic categories of tests:

- External stored response testing. This is the most common form for today's integrated circuits and it relies heavily on large IC **automated test equipment (ATE)**.
- **Builtin-self-test (BIST).** This is method of defining and building test circuity onto the integrated circuit. This technique provides stimulus patterns and observes the logical output of the integrated circuit.
- **Scan** testing. Some or all of the sequential elements are converted into a shift register for control and observation purposes.
- Parametric tests. The values of circuit parameters are directly measured. This includes the I_{DDQ} test, which is good at detecting certain classes of defects. To implement an I_{DDQ} test, a CMOS integrated circuit clock is stopped and the quiescent current is measured. All active circuitry must be placed in a low-current state, including all analog circuitry.

External Stored Response Testing

Figure 14.4 shows a typical integrated circuit being tested by a stored response tester. Note that the patterns for the input and output of the integrated circuit are applied by primary inputs, and the output is compared to known good response. The process of generating stored response patterns is usually done by simulation.

© 2006 by Taylor & Francis Group, LLC

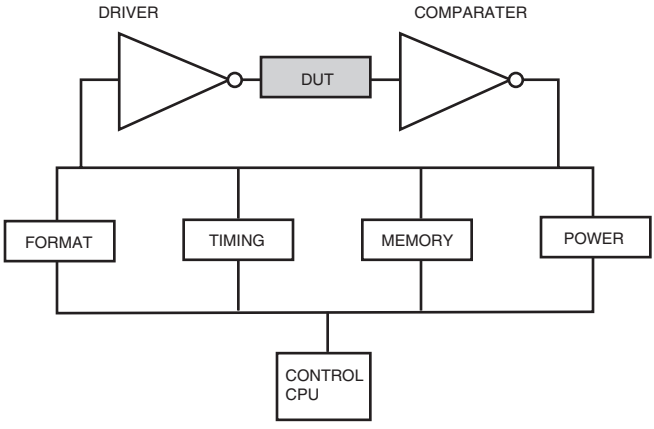


FIGURE 14.4 Stored response tester and DUT.

Often these patterns are trace files from the original logic simulation of the circuit. These files may be used during simulation for correct operation of the logic and later transferred to a tester for debug, circuit verification, and test. The patterns may take up considerable storage space, but the test pattern can easily detect structural and/or functional defects in an integrated circuit.

To exercise the **device under test** (**DUT**), a stored response functional tester relies on patterns saved from the logic simulation. These patterns include input and output states and must account for unknown or X states. The set of patterns for functional test can usually be captured as trace files, or change files, and then ported to the test equipment. Trace or change files capture the logic state of the DUT as its logical outputs change over time. Typically, these patterns represent the logical operation of the device under test. As such, they are good for debugging the operation and design of integrated circuits. They are not the best patterns for fault isolation or yield analysis of the device.

BIST

The most common implementation of BIST is to include a **linear feedback shift register** (**LFSR**) as an input source. An LFSR is constructed from a shift register with the least significant bit fed back to selected stages with exclusive-OR gates. This feedback mechanism creates a polynomial, which successively divides the quotient and adds back the remainder. Only certain polynomials create pseudorandom patterns. The selection of the polynomial must be carefully considered. The LFSR source, when initialized and clocked correctly, generates a pseudorandom pattern that is used for the stimulus patterns for a block of logic in an integrated circuit.

The common method of output compaction is to use a **multiple input shift register** (**MISR**). A MISR is an LFSR with the outputs of the logic block connected to the stages of the LFSR with additional exclusive-OR gates. The block of logic, if operating defect free, would give a single correct signature (see Fig. 14.5). If there is a defect in the block of logic, the resultant logic error would be captured in the output MISR. The MISR constantly divides the output with feedback to the appropriate stages such that an error will remain in the MISR until the results are read at the end of the test sequence. Because the logic states are compacted in a MISR, the results may tend to mask errors if the error repeats itself in the output, or if there are multiple errors in the block. Given a number of errors, the output could contain the correct answer even though the logic block contains errors or defects. This problem is called *aliasing*. For example, the output register shown in Fig. 14.5 is 20-bits long; 2^{20} is approximately one million, which means that there is a very high chance that without an aliasing defect, the output will be the correct state of one out of a million with a fault free integrated circuit.

© 2006 by Taylor & Francis Group, LLC

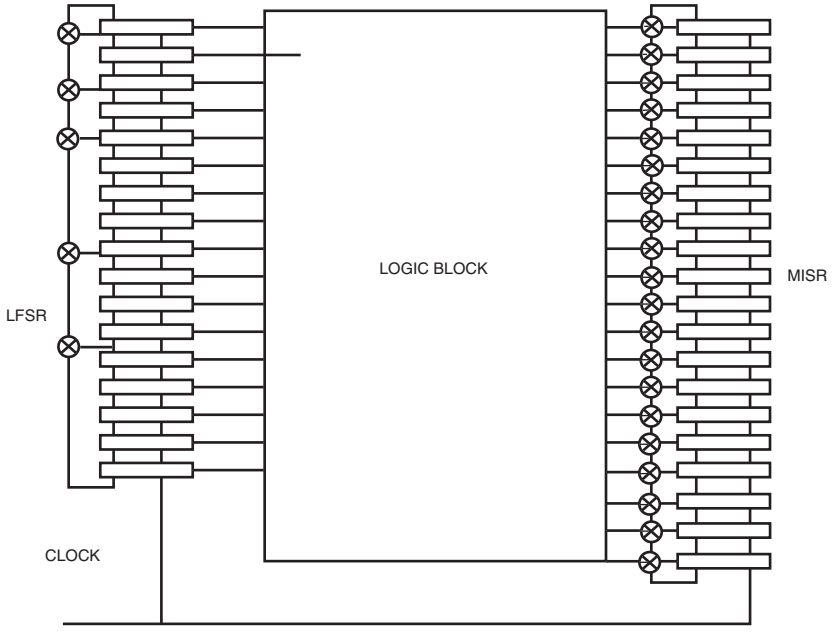


FIGURE 14.5 BIST implementation example with inputs and outputs.

It should be noted that the example is shown for combination logic only. Sequential logic becomes more difficult as initialization, illegal states, state machines, global feedback, and many other cases must be accounted for in the generation of BIST design and patterns.

Scan

Scan is a technique where storage elements (latches and flip-flops) are changed to dual mode elements. For instance, Fig. 14.6 shows a normal flip-flop of an integrated circuit. Also shown in Fig. 14.6 is a flip-flop converted to a scan register element. During normal operation, the D input and system clock control the output of the flip-flop. During scan operation, the scan clocks are used to control shifting of data into

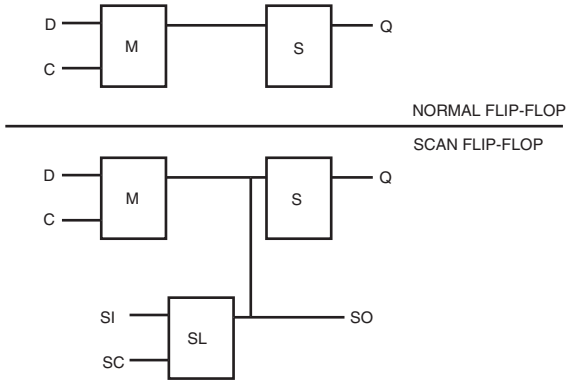


FIGURE 14.6 Example of a scan latch and flip flop.

© 2006 by Taylor & Francis Group, LLC

and out of the shift register. Data is shifted into and out of the master and scan latch. The slave latch can be clocked by the system clock in order to generate a signal for the required stimulation of the logic between this scan latch and the next scan latches. Figure 14.7 shows latches and logic between latches. This method successfully reduces the large sequential test problem to a rather simple combinatorial problem. The unfortunate problem with scan is that the area overhead is, by far, the largest of all **design for test (DFT)** methods. This area increase is due to the fact that each storage element must increase in transistor count to accommodate the added function of scan.

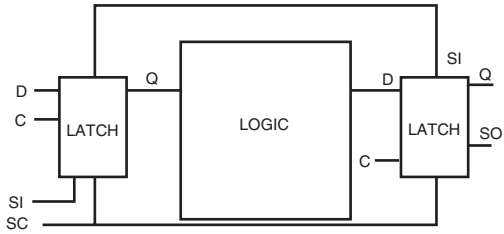


FIGURE 14.7 Use of scan latch for logic.

Note that the scan latch contains extra elements for control and observation. These extra elements include the scan-clock (SC), scan-in (SI), and scan-out (SO). As shown in Fig. 14.7, the scan-clock is used to drive all scan latches. This controls the shift register function of the scan latch, and will allow a known one or zero to be set for each latch in the scan chain. For each scan clock applied, a value of one or zero is placed on the scan-in line, which is then shifted to the scan-out and the next scan-in latch. This shifting stops when the entire chain is initialized. At the end of the scan in sequence, a system clock is triggered, which launches data from the scan portion of the latch of the Q output. Then it goes through the combinational logic to the next stage D input where another system clock stores it in the latch. The next set of scan loads, and shifts out the stored value of the logic operation for comparison. If the logic operation was correct, the scan value will be correct in the shift out sequence.

I_{DDQ}

I_{DDQ} is perhaps the simplest test concept. Figure 14.8 shows a typical dual inverter structure with a potential defect. It also shows the characteristic of I_{DDQ}. The upper circuit is defect free, and the lower circuit has a leakage defect shown as R. I_{DDQ} is shown for each clock cycle. Note the elevated current in the defective circuit during clock cycle A. This elevated current is called I_{DDQ} . It is an easy way to check for most bridging and many other types of defects. The I_{DDQ} test controls and observes approximately half of the transistors in a IC for each test. Therefore, the test is considered massively parallel and is very efficient at detecting bridge defects and partial shorts. Another consideration is the need to design for an I_{DDQ} test. All DC paths, pullups, bus drivers, and contention paths must be designed for zero current when the clock is stopped. One single transistor remaining on, can disable the use of an I_{DDQ} test.

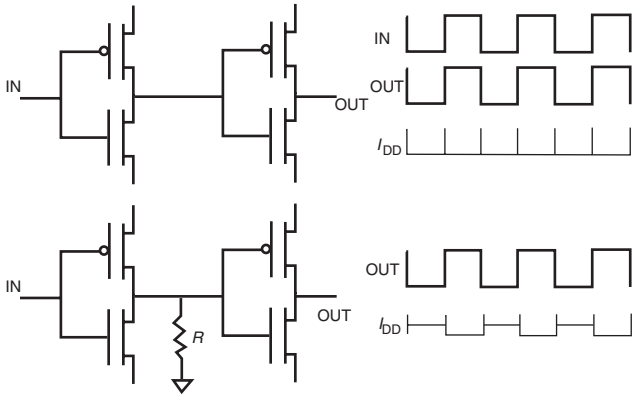


FIGURE 14.8 I_{DDQ} test values.

© 2006 by Taylor & Francis Group, LLC

TABLE 14.2 Attributes of Test

Test Types	Complexity	Illegal States	Aliasing	Silicon Overhead	Test Pattern Generation Time
SCAN	Simple	Controlled	NA	Highest	Shortest
BIST	Complicated with latches	Big problem	Yes	Moderate	Moderate
Stored response	Very complicated	NA	NA	None-little	Longest
I_{DDQ}	Low	None	None	None	Simple

Note that V_{out} and I_{DD} in the defective case may be tolerable if the output voltage is sufficient to drive the next input stage. Not shown here is the timing impact of the degraded voltage. This degraded voltage will cause a time delay in the next stage.

14.4 Test Tradeoffs

To determine the best type of test, one must understand the defect types, the logical function of the integrated circuit, and test requirements. Examples of test tradeoffs are included in Table 14.2. The rows in the table correspond to the test types as listed earlier: scan, BIST, stored response, and I_{DDQ}. The columns in the table depict certain attributes of test to be described here.

The first column is the test type. Column number two shows pattern complexity. This is the problem of generating the necessary patterns in order to fully stimulate and control the nodes of the integrated circuit. Column number three is illegal states. Often times the logic in integrated circuits, such as bidirectional bus drivers and signals, must be exclusively set one way or another. Examples include decoders, bus drivers and mutually exclusive line drivers. Tests such as scan and BIST, if not designed and implemented correctly, can damage the integrated circuit during the test process by putting the circuit into an internal contention state. Examples include two bus drivers on simultaneously when one drives high and the other drives low.

Aliasing is a common problem of BIST techniques. This is shown in column four. For example, consider a memory laid out in rows and columns and assume that one column is a complete failure. If there are multiple failures along the column, let us say 256, and if the signature length is 256, the output could possibly show a correct response even though the entire column or row of the memory is defective.

The next column shows overhead. Overhead is the amount of silicon area necessary to implement the test. Each of the areas of overhead relate to the amount of silicon area needed to implement the specific types of test methods.

The final column is the time for test generation. The unfortunate relationship here is that the techniques with the highest area overhead have the most complete control and are easiest to generate tests. Techniques with the lowest area needs have the most complex test generation problem, and take the longest time. This is a complex trade off.

Tradeoffs of Volume and Testability

It is important to forecast the volume of the integrated circuit to be manufactured prior to selecting DFT and test methods. Engineering effort is nonrecurring and can be averaged over every integrated circuit manufactured. The testing of an integrated circuit can be a simple or a complex process. The criteria used for selecting the proper test technique must include volume, expected quality, and expected cost of the integrated circuit. Time to market and/or time to shipments are key factors, as some of the test techniques for very large integrated circuits take an extremely long time.

Defining Terms

Automated test equipment (ATE): Any computer controlled equipment used to test integrated circuits.

Builtin self-test (BIST): An acronym that generally describes a design technique with input stimulus generation circuitry and output response checking circuitry.

© 2006 by Taylor & Francis Group, LLC

Defect: Any error, particle, or contamination of the circuit structure. Defects may or may not result in faulty circuit operation.

Design for test (DFT): A general term which encompasses all techniques used to improve control and observation of internal nodes.

Device under test (DUT): A term used to describe the device being tested by the ATE.

Fault: The operation of a circuit in error. Usually faulty circuits are the result of defects or damage to the circuit.

Linear feedback shift register (LFSR): A method of construction of a register with feedback to generate pseudo random numbers.

Multiple input shift register (MISR): Usually an LFSR with inputs consisting of the previous stage exclusive OR'ed with input data. This method compacts the response of a circuit into a polynomial.

Scan: A design technique where sequential elements (latches, or flipflops) are chained together in a mode that allows data shifting into and out of the latches. In general, scan allows easy access to the logic between latches and greatly reduces the test generation time and effort.

References

Abramovici, M. et al. 1990. *Digital Systems Testing and Testable Design*. IEEE Press, New York.

Needham, W.M. 1991. *Designer's Guide to Testable ASIC Devices*. Van Nostrand Reinhold, New York.

van de Goor, A.J. 1991. *Testing Semiconductor Memories, Theory and Practice*. John Wiley & Sons, New York.

Further Information

IEEE International Test Conference (ITC). This is the largest gathering of test experts, test vendors and researchers in the world.

IEEE Design Automation Conference (DAC) held in various sites throughout the world.

IEEE VLSI Test Symposium.

Test vendors of various types of equipment provide training for operation, programming, and maintenance of ATE.

CAD vendors supply a variety of simulation tools for test analysis and debugging.

© 2006 by Taylor & Francis Group, LLC

15

Semiconductor Failure Modes[1]

15.1 Discrete Semiconductor Failure Modes **15**-1
15.2 Integrated Circuit Failure Modes **15**-5
15.3 Hybrid Microcircuits and Failures **15**-7
15.4 Memory IC Failure Modes **15**-7
15.5 IC Packages and Failures **15**-8
15.6 Lead Finish **15**-12
15.7 Screening and Rescreening Tests **15**-13
15.8 Electrostatic Discharge Effects **15**-14

Victor Meeldijk

15.1 Discrete Semiconductor Failure Modes

Failures of semiconductor devices in storage, or dormant applications, are the result of latent manufacturing defects that were not detected during device screening tests. For discrete semiconductors, such as transistors, a large percentage of failures are the result of **die** and **wire bonding** defects and contamination. A common failure mode of spring loaded diodes is the contact material losing its compression strength, or slipping off the die, resulting in an open circuit.

Failure mechanisms can be grouped into three categories:

- Those independent of environment (oxide defects, diffusion defects).
- Failures dependent on environment (wire bond or **metallization** defects, see Fig. 15.1 and Fig. 15.2).
- Failure mechanisms that are time and environment dependent (metal migration, corrosion, intermetallic compound formulations, such as caused by dissimilar metal use).

Table 15.1 and Sec. 15.6 discusses the various screening tests to induce failures prior to components being incorporated in assemblies.

Common failure modes of discrete semiconductors are zener diode: 50% short, 50% open; junction diode: 60–70% high reverse, 20–25% open, 10–15% short; silicon controlled rectifier (SCR): 2% open, 98% short; and transistor: 20% high leakage, 20% low gain, 30% open circuit, 30% short circuit (Fig. 15.3).

[1] Portions of this chapter were adapted from: Meeldijk, V. 1995. *Electronic Components: Selection and Application Guidelines,* Chaps. 10 and 11. Wiley Interscience, New York. Used with permission.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

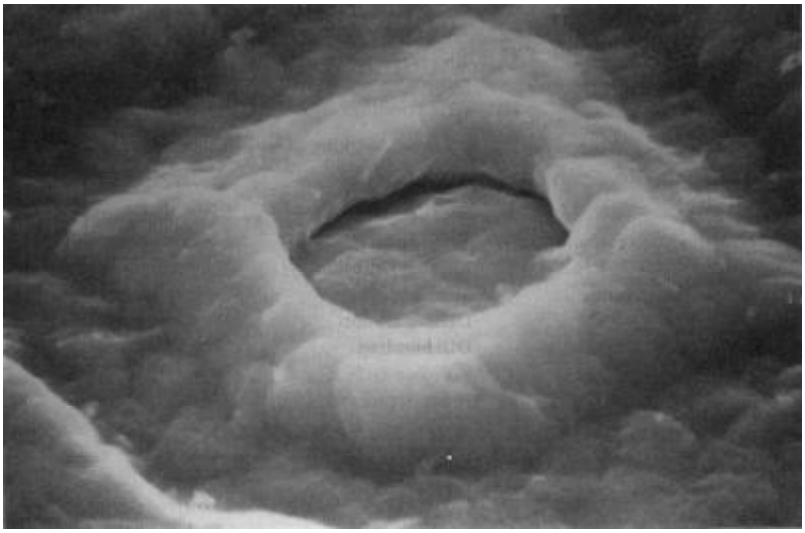


FIGURE 15.1 Excessive thinning or metallization.

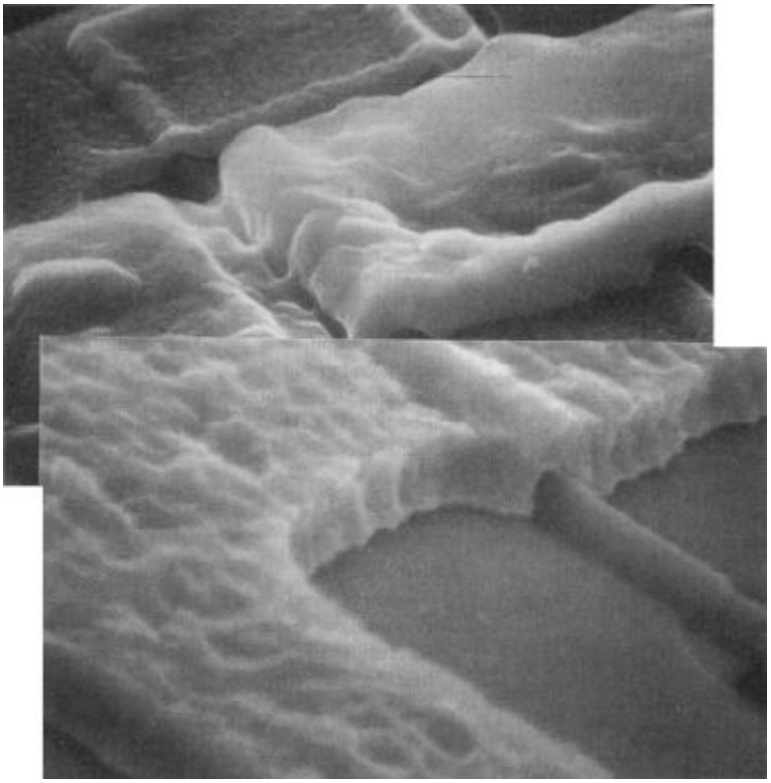


FIGURE 15.2 Separation/depression of metallization and thinness of metallization. The second photograph shows an acceptable metallization step.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

TABLE 15.1 Various Screening Tests Used to Detect Defects

Test	Process Defect	Failure Mode
Electrical: DC tests, static/dynamic tests, functional tests, Schmoo plots	Slice Preparation Etching Wire Bonding	Improper electrical performance, opens, shorts, degraded junction characteristics, intermittents
	Improper marking	Device operates differently than expected, or not at all
Thermal cycling: MIL-STD-883 (test) method 1010; Japanese Industrial Standards Office (JIS) C 7022 A-4; International Electrotechnical Commission (IEC) PUB 68; Test Na, Nb	Slice preparation **Passivation** (Fig. 15.7) Etching; Diffusions Metallization **DIE separation** **DIE bonding** Wire bonds; Final seal Fatigue cracks	Potential short or open circuits, low breakdown voltage, increased leakage, poor performance due to faulty components, near shorts, near opens, high resistance internal connections
(Temp/humidity cycling MIL-STD-883 method 1004; JIS C 7022 A-5; IEC PUB 68; Test Z/AD)		
High temperature storage: MIL-STD-883, method 1008; JIS C 7022, B-3	Etching Slice preparation Diffusions; Metallizations Lead bonds; Final seal Creep; Corrosion Electromigration; Surface charge spreading	Lowered device performance due to degradation of junction (soldering characteristics, shorts, potential shorts, opens, high resistance)
High temperature/high humidity bias test and storage test, JIS C 7022, B-5; IEC PUB 68; Test C (heat test, MIL-STD-750, method 2031; JIS C 7022; A-1; IEC PUB 68; Test Tb)		
(High temperature operation, MIL-STD-883 method 1005; JIS C 7022 B-1)		
(Low temperature storage, JIS C 7022, B-4; IEC PUB 68; Test A)		
Reverse bias voltage:	Slice preparation Final seal	Degradation of junction characteristics, performance degradation
Operational life test:	Slice preparation Passivation; Etching Diffusions; Metallization Die bonding; Final seal Dendrite growth	Degradation of junction characteristics, short circuits, leakage, low insulation resistance, performance degradation from heating or faulty or unstable internal components, opens, near opens, near shorts, high resistance internal connections
Mechanical: Variable freq. vibration MIL-STD-883 method 2007; JIS C 7022; A-10; IEC PUB 68; Test Fc	Die separation Die bonding; Wire bonds; Die cracks; Lead shape; irregularities package; irregularities creep	Performance degradation caused by overheating, opens or potential opens, shorts or intermittent shorts
Solderability: MIL-STD-883; METHOD 2003; JIS C 7022; A-2;	Lead finish/plating	Intermittent, open or high resistance solder connections between the part and the circuit board
Salt fog (atmosphere) MIL-STD-883 method 1009; JIS C 7022; A-12; IEC PUB 68; Test Ka		Tests resistance to corrosion in a salt atmosphere

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

TABLE 15.1 Various Screening Tests Used to Detect Defects (*Continued*)

Test	Process Defect	Failure Mode
Thermal shock: MIL-STD-883; method 1011; JIS C 7022; A-3; IEC PUB 68; Test Nc	Die separation Die bonding	Performance degradation caused by overheating, opens or potential opens, shorts or intermittent shorts
X Ray:	Die bonding; Wire bonding Final seal Lead frame shape irregularities	Performance degradation caused by overheating, intermittents, shorts or opens, intermittent short circuits caused by loose conductive particles in the package.
Acceleration: Constant acceleration; MIL-STD-883 method 2001; JIS C 7022; A-9; IEC PUB 68; Test Ga	Die bonding Final seal	Performance degradation caused by overheating, weak wires will be open or intermittent, intermittent shorts caused by loose conductive particles in the package
Hi-voltage test:	Passivation Final seal	Open or short circuits, increased device leakage, low insulation breakdown
ESD testing: MIL-STD-883; method 3015 Electronic Industries Association of Japan (EIAJ) IC 121:20	Design susceptibility to ESD	Degraded device performance or device failure.
Vibration/PIND test:	Die bonding Wire bonding	Shorts or intermittent shorts caused by loose conductive particles in the package, performance degradation caused by overheating, opens and intermittents
Lead fatigue: Lead integrity; JIS C 7002; A-11; IEC PUB 68; Test U	Final seal	Open circuits, external lead loose
Leak tests: Hermetic seal tests; MIL-STD-883 method 1014; JIS C 7022; A-6; IEC PUB 68; Test Q [Pressure cooker test (PCT) to evaluate moisture resistance in a short period of time; EIAJ IC-121: 18]	Final seal	Performance degradation, shorts or opens caused by chemical corrosion or moisture, intermittents
Precap visual:	Slice preparation Passivation; Masking Etching; Metallizations Die separation Die bonding; Wire bonding	Increased leakage, low insulation breakdown, opens, shorts, intermittents, potential shorts or opens, high internal resistances, intermittent shorts
Visual:	Final seal	Open circuits; package cracks; package seal problems

Interpretation of failure data by the Reliability Analysis Center (RAC, 1991), Rome, New York (used with permission), has the following failure mode breakdowns:

- General purpose diode: 49% short, 36% open, 15% parameter change
- Diode rectifier: 51% short, 29% open, 20% parameter change
- Small signal diode: 18% short, 24% open, 58% parameter change
- Microwave diode: 50% open, 23% parameter change (drift), 10% short, 17% intermittent
- Zener diode reference: 68% parameter change (drift), 17% open, 13% shorted, 2% intermittent
- Zener diode regulator: 45% open, 35% parameter change (drift), 20% shorted

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

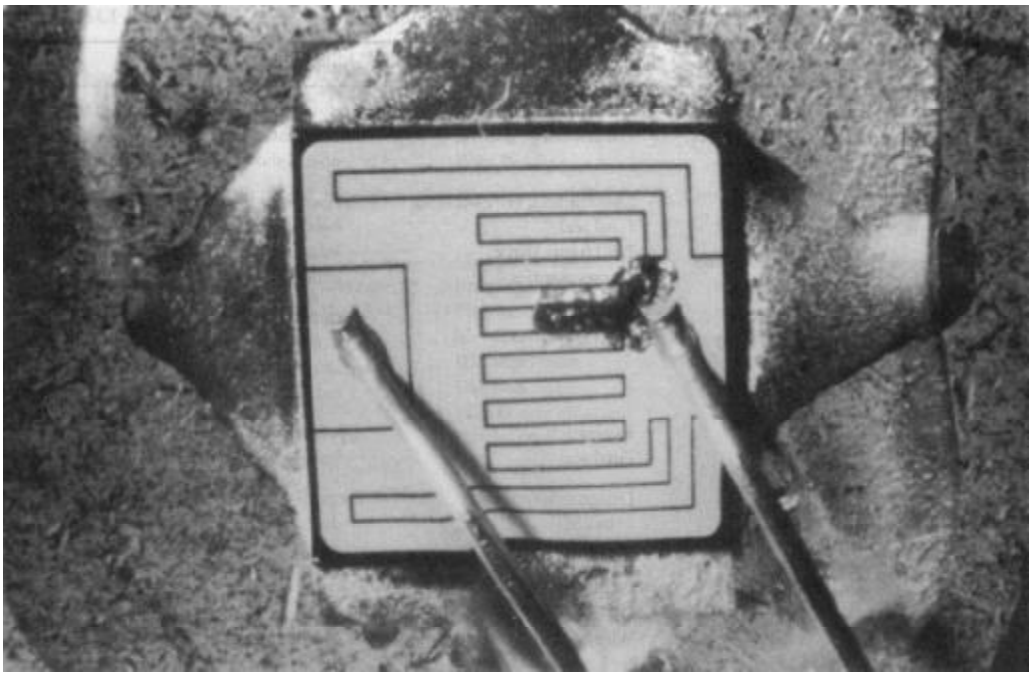


FIGURE 15.3 A failed transistor.

- Optoelectronic LED: 70% open, 30% short
- Optoelectronic sensor: 50% short, 50% open
- Thyristor: 45% failed off, 40% short, 10% open, 5% failed on
- Triac: 90% failed off, 10% failed on

Further breakdown on transistor failure modes by the Reliability Analysis Center (RAC) are

- Bipolar transistor: 73% shorted, 27% open

(Note these new data contrast with that published in January 1976 by the U.S. Army Material Command (AMCP-706-196), which had 59% high collector to base leakage current, 37% low collector to emitter breakdown (Bvceo), and 4% open circuited).

- Field effect transistor (FET): 51% shorted, 22% low output, 17% parameter change, 5% open, 5% output high
- GaAs FET transistor: 61% open, 26% shorted, 13% parameter change
- Radio frequency (RF) transistor: 50% parameter change, 40% shorted, 10% open

15.2 Integrated Circuit Failure Modes

Most IC failures, as for semiconductors, are related to manufacturing defects (Fig. 15.4 is the inside view of a typical acceptable IC). A breakdown of typical IC failure mechanisms is 40.7% wire bond interconnects, 23.4% misapplication/misuse, 4.2% **masking**/etching defect, 3.3% die mechanical damage, 1.4% cracked die, 0.9% die metallization corrosion, 0.9% die contamination, and 24.8% other causes.

These mechamisms result in the following typical failure modes:

- Digital devices: 40% stuck high, 40% stuck low, 20% loss of logic
- Linear devices: 10–20% drift, 10 output high or low, 70–80% no output

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

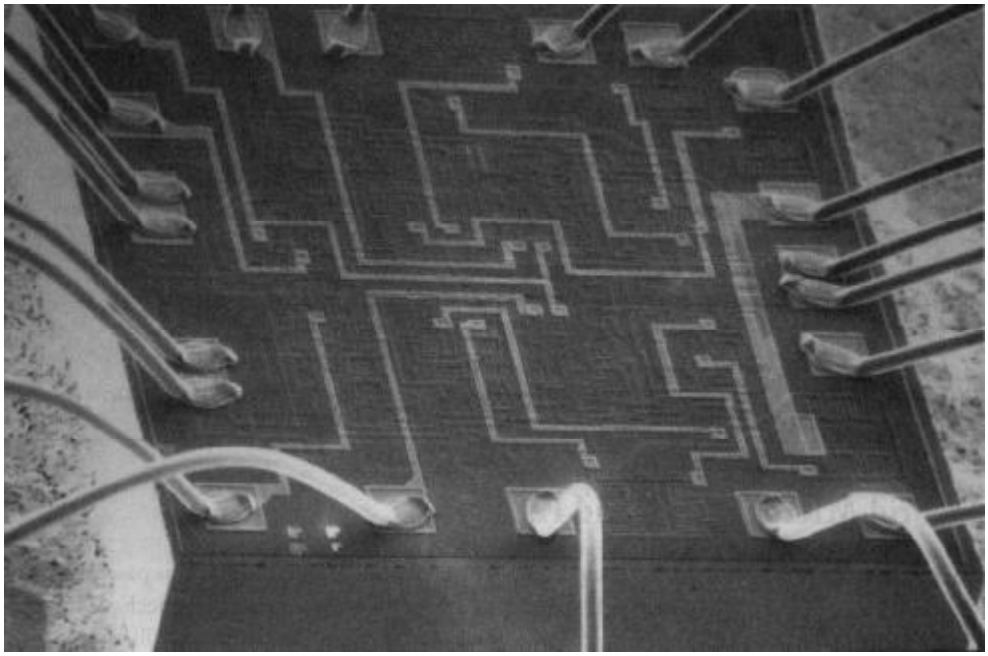


FIGURE 15.4 Inside view of an acceptable IC showing **bond pads** and metallization.

Interpretation of failure data in 1991 by the Reliability Analysis Center (RAC) found the following failure mode percentages:

- Digital bipolar: 28% output stuck high, 28% output stuck low, 22% input open, 22% output open
- Digital MOS: 8% output stuck high, 9% output stuck low, 36% input open, 36% output open, 12% supply open
- Digital programmable array logic (PAL): 80% failed truth table, 20% shorted
- Interface IC: 58% output stuck low, 16% input open, 16% output open, 10% supply open
- Linear IC: 50% degraded/improper output, 41% no output, 3% shorted, 2% open, 2% drift
- Linear-op amps: 69% degraded (unstable, clipped output, drifted etc.), 13% intermittent, 10% shorted, 6% overstressed by transients, 3% no output
- Bipolar memory: 79% slow transfer of data, 21% data bit loss
- MOS memory: 34% data bit loss, 26% short, 23% open, 17% slow transfer of data
- Digital memory: 30% single bit error, 25% column error, 25% row error, 10% row and column error, 10% complete failure
- Digital memory RAM: 25% no operation at cold temperatures, 17% parameter change, 16% shorted, 13% open, 13% incorrect data, 7% contaminated (particles inside cavity of hermetic sealed unit)
- Ultraviolet erasable programmable read only memory (UVEPROM): 94% open (unprogrammable) bit locations, 6% would not erase
- Hybrid devices: 51% open circuit (caused by resistor/capacitor opens, internal bond pad corrosion and electromigration), 26% degraded/improper output (distorted or slow response time output), 17% short circuit (**electrostatic discharge (ESD)** overstress, fractures, shorting wire bonds), 6% no output

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

TABLE 15.2 Hybrid Microcircuits—Failure Accelerating Environments

Accelerating Environment	Environments Failure Mechanism	Failure Mode
Mechanical stress (thermal shock, vibration)	Substrate bonding; Cracked/broken substrate; Faulty bonds	Open circuits
High temperature	Damaged resistors	Open or out of tolerance
Thermal cycling	Cracked resistors; Various film defects Excess bonding time Thermal coefficient of expansion mismatch between film and substrate	Open or out of tolerance
High voltage and temperature test	Various film defects; Excess bonding time Thermal coefficient of expansion mismatch between film and substrate	Out of tolerance
Thermal and mechanical stress	Cracked dice; Shorted wires Faulty bonds; Bond separation	Open circuits; Short circuits

The analysis data, which can be interpreted by the reader for various applications, is in their report (RAC, 1991). Most vendors have reliability reports on their products that discuss the results of accelerated life tests and provide a device failure rate.

15.3 Hybrid Microcircuits and Failures

Hybrid devices are a combination of active components (microcircuits or ICs) and various passive discrete parts mounted on a substrate and interconnected by conductive film traces. Hybrid failure mechanisms, like those of discrete and integrated circuits, are primarily due to manufacturing defects. Failures will occur whether the device is operating or dormant. Various accelerating environments, such as temperature or vibration can reveal the failure mechanisms before parts are shipped (or used in circuits). Table 15.2 lists the accelerating environments and the problems they reveal.

15.4 Memory IC Failure Modes

Memory ICs failures result in the inability to read or write data, erroneous data storage, unacceptable output levels, and slow access time. The specific failures that cause these problems are as follows:

- Open and short circuits: can cause various problems from a single bit error to a catastrophic failure of the whole device.
- Open decoder circuits: cause addressing problems to the memory.
- *Multiple writes:* writing one cell actually writes to that cell and other cells (multiple write and address uniqueness problems).
- Pattern sensitivity: the contents of a cell become complemented due to read and write operations in electrically adjacent cells (cell disturbances, adjacent cell disturbances, column disturbances, adjacent column disturbances, row disturbance, adjacent row disturbance).
- *Write recovery:* the access time of the device may be slower than specified when each read cycle is preceded by a write cycle, or a number of write cycles.
- Address decoder switching time: this time can vary depending on the state of the address decoder prior to switching and on the state to which the decoder is switching.
- Sense amplifier sensitivity: memory information may be incorrect after reading a long series of similar data bits followed by a single transition of the opposite data value.
- *Sleeping sickness:* the memory loses information in less than the specified hold time (for DRAMs).

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

- Refresh problems (may be static or dynamic): the static refresh test checks the data contents after the device has been inactive during a refresh. In dynamic refresh, the device remains active and some cells are refreshed. All cells are then tested to check whether the data in the nonrefreshed cells is still correct.

Special tests are needed for memory devices to insure the integrity of every memory bit location. These functional tests are composed of four basic tests, pattern tests, background tests, timing test, and voltage level tests. A summary of these tests is contained in Table 15.3.

Memory tests cannot be specified to test each part 100% as a RAM can contain any one of 2^N different data patterns and can be addressed in N factorial ($N!$) address sequences without using the same address twice. Test times shown may be extremely long for some high-density memory devices; a GALPAT test for a 4 M DRAM would take 106 days to execute. It is easy to see how impractical it is to overspecify device testing. Any test plan should be developed along with the supplier of the memory devices you are to purchase.

System level tests may use bit-map graphics where the status of the memory cells is displayed on the cathode ray tube (CRT) display of the system (i.e., failing bits show up as a wrong color). Various patterns are read into the memory and the CRT may be configured to show 1 cell equivalent to one pixel (a picture element) or compressed to show 4, 16, or 64 cells per pixel depending on the number of pixels available and the memory size. Zoom features can be used to scale the display for close examination of failing areas of the memory.

Note, in large memory intensive systems, consider using error correction codes to correct hard (permanent failure) and soft errors (i.e., alpha particle upsets, a single upset that does not occur again). One vendor of video RAMs estimates a soft error rate of a 1 Meg part of 3.9 FITs (at a 500-ns cycle time with Vcc at 4.5 V, and a 4 Meg DRAM error rate of 41 FITs (at a 90% confidence level, 5 Vcc operation with 15.625-μs cycle (refresh) rate). Soft error rates are dependent on cycle time and operating voltage; the lower the voltage and faster the cycle time, the higher the failure in time rate. One thousand FITs equals 1 PPM (1 failure in 10^6 h), which equals 0.1%/1000 (0.1% failures every 1000 h). When calculating soft error rates, add the refresh mode rate and the active mode rate, which can be determined from acceleration curves provided by the manufacturer.

15.5 IC Packages and Failures

Devices can be classified by package style, which can either be **hermetically** sealed (**ceramic** or metal cans) or nonhermetic (epoxy, silicones, phenolics or plastic encapsulated). Materials employed in most microcircuits will change very slowly if stored in a dry environment at a constant low temperature.

Hermetic packages, glass, ceramic, or metal, protect the device from environmental atmospheric effects. Depending on the metal package it can take anywhere from 50 to well over 100 years for the inside of the IC package to reach 50% of the humidity of the outside air. An epoxy package can take anywhere from minutes to a few days. Therefore, equipment designed for harsh environments, such as might be encountered in military applications, use hermetically sealed devices. Plastic packaged devices are low cost, offer mechanical shock and vibration resistance, are free from loose particles inside the device as they do not have a cavity, and are available in smaller package sizes. All plastics, however, contain some moisture and are permeable to moisture to some degree. (Salt environments have also been shown to effect silicone encapsulated devices but leave epoxy novalac parts relatively unaffected). The mismatch in thermal coefficients of the plastic vs. the die also reduces the operating temperature of the device from 0 to 70°C vs. hermetic packages, which can operate from −55 to +125°C.

Studies have been done by the Microelectronics and Computer Technology Corporation (Austin, Texas) and Lehigh University on replacing heremetic packages with more effective lightweight protective coatings. Their program, *reliability without hermeticity* (RwoH), was initiated by a research contract awarded to them by the U.S. Air Force Wright Laboratory. The RwoH is an industry working group, initiated in

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

TABLE 15.3 Tests for Memory Circuits

Test	Comments
March	An N pattern test that checks for functional cell failures (opens and shorts and address unique faults) and data sensitivity. In this test a background of 0s is written into memory. Then a 1 is written into each sequential cell. Each cell is then tested and changed back to 0 in reverse order until the first address is reached. The same sequence is then repeated by loading in a background of 1s.
Masest	An N pattern test that checks for address uniqueness and address decoder problems (open decode lines). This is an alternating Multiple-Address Selection test that writes a background of 0s to the memory device at the min address, then the max address, then the min $+1$ address, then the max -1 address, etc. All cells are then read and verified and a complementary pattern of 1s is read into the memory device and the test is repeated.
Refresh	Any one of several patterns designed to test for an acceptable refresh time. Usually an N pattern test
Sliding diagonal	An $N^{3/2}$ pattern that checks for address decoder problems, sense amplifier faults, and sense amplifier recovery. This test is a pattern that generates a diagonal of data in a field of complement data, and slides the diagonal of data in the x direction, creating a barber pole effect.
Surround disturbance	An N pattern test that checks for adjacent cell disturbance. Variations include a row disturbance and column disturbance.
Galloping tests	This is a sequence test where a base cell, containing complement data from the rest of the cells in its set (i.e., the base cell is 1 but the other cells are 0). A set may be a column, row, device or adjacent cells to the base cell.
Walking tests	Walking tests are similar to galloping tests except that the cells in the set with the base cell are read sequentially instead of alternately with the base cell.
Galloping/walking tests:	
GALPAT rows (galloping rows) Rows	An $N^{3/2}$ pattern test that checks adjacent cell walking disturbance and row/column disturbance.
GALPAT columns (galloping columns) Walking columns	An $N^{3/2}$ test that checks for row/column disturbances.
GALPAT (galloping pattern test)	A $4N^2$ pattern test that checks for address decoder problems, adjacent cell disturbances, row/column disturbances and sense amplifier recovery. This is a long execution time test. Into a background of 0s the first cell is complemented (to 1) and then read alternately with every other cell in memory. This sequence is repeated until every memory cell is tested. The test is then repeated with a background of 1s with the cell changed to 0.
GALDIG (galloping diagonal) pattern test	This test finds unsatisfactory address transitions between each cell and the positions in the cell's diagonal. It also finds slow sense amplifier recovery and destruction of stored data due to noise coupling among cells in a column. In this test a diagonal of 1s is written into a background of 0s. All cells are read and verified and the diagonal is then shifted horizontally. This keeps repeating until the diagonal has been shifted throughout the memory. The test is then repeated with complemented data.
GALWREC	A galloping write recovery test, an N^2 pattern test

(*continued*)

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

TABLE 15.3 Tests for Memory Circuits (*Continued*)

Test	Comments
WALKPAT (walking pattern)	An N^2 test that checks for address uniqueness and sense amplifier recovery. This is a long execution time test. In a background of 0s, the first cell is complemented and then all the other cells are read sequentially. The cell is then changed back to zero and the next cell is complemented and all the other cells are then read sequentially again. This repeats for all of the memory cells and then repeats again with a background of 1s with each cell changed to a 0. (A single 1 moves through a field of 0s and a single 0 is moved through a field of 1s.)
Dual walking columns	An $N^{3/2}$ pattern test
MOVI (moving inversion)	This test checks for address propagation failures. It is similar to a march but in addition uses each address line as the least significant bit in a counting sequence.
Scan test	Each cell is written, or read, sequentially. The addressing sequence can count up from the minimum address (minmax) or down from the maximum address (maxmin). Either the x or y axis can be the fast axis (rowfast or columnfast, respectively).
Address complement	An addressing sequence in which all address lines, except one, are changed in each cycle.
Background tests (may be used with the preceding tests)	
Checkerboard inverted checkerboard	An N pattern test that checks for functional cell failures and data sensitivity (to noise). Each cell is checked to verify that it can be forced to a 1 or a 0. Each cell is surrounded by cells with complementary data (a logical 1 stored in a cell would be surrounded by cells that have logical 0 stored in them). Variations include double checkerboard and complemented double checkerboard which have the same execution time.
Pseudocheckerboard	This checkerboard pattern is generated by exclusive OR-ing data with the least significant row address bit and the least significant column address bit (data XOR X0 XOR Y0). This may not generate a true checkerboard background in parts that have folded digit lines, and so a different equation may be necessary for these parts.
Zeros/ones	An N pattern test. A background of either solid 1s or 0s is stored in each cell.
Row stripes	Background data where every row is surrounded by its complement. Therefore a row with logical 1 stored in it would be surrounded by rows with logical 0 in them.
Column stripes	Like row stripes but background data relates to columns instead of rows.
Double variations	The size of the background is doubled, two rows, two columns, or a double checkerboard with two rows by two columns.
Parity	Background data that are generated when the parity of the x and y addresses is even or odd.
Bit map graphics (test that can be used to check memory at system level)	Used to examine patterns of failing cells, characterize the operating regions of memory devices, verify device specifications. This test identifies mask misalignment, diffusion anomalies, contamination defects, design sensitivities and semiconductor defects.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

1988, to address nonhermetic electronic packaging issues, specifically, coating to provide environmental protection for semiconductor devices and multichip modules.

The Joint Electron Device Engineering Council (JEDEC) specification A112 defines six levels of moisture sensitivity with level 1 being the highest rating, or not moisture sensitive. The other levels are reduced levels, such as level 3, which ensures product integrity (i.e., from **popcorning** from entraped moisture during soldering) when stored at 30°C, 85% Rh for 168 h. Level 4 indicates package integrity is ensured for 84 h, and level 6 only permits 6 h of exposure. Plastic parts shipped in sealed bags with desiccant usually are designed for a 12-mo storage and should only be opened when the parts are to be used. Parts stored for longer than this time, especially plastic quad flatpack (PQFP) packaged devices, should be baked to remove the moisture that has entered the package. Entrapped moisture can vaporize during rapid heating, such as in a solder reflow process, and these stresses can cause package cracking. Subsequent high-temperature and moisture exposures can allow contaminants to enter the IC and cause failure at a later time due to corrosion.

Note Hughes Missile Systems reported in 1995 of conditions where organic-based (such as organic acid (OA)) water-soluble fluxes lead to dendrite growth and short circuits. It was found that component seals of high lead glass **frit** (glass frit is used in cerdip and ceramic flat pack components) and tin and lead on the device **leads** results in the growth of lead dendrites on the surface of the glass seals. The lead comes from the high lead oxide content of the soft glass itself. High temperatures accelerate the dendrite growth, with shorts occurring in as little as 5 min with 140°F warm flux. Prevention of this problem includes using different fluxes and minimizing the application of fluxes to the component side of the PC board. Dendrite growth has also been reported for kovar flatpacks stored in foil-lined cardboard packaging. The chlorine dendrite growth, resulting in a high electrical leakage from pins to case, resulted from the chemical impurities in the cardboard and the device. Fiberboard and paperboard packaging is no longer used for device unit packaging. **Tin whisker** growth also occurs with parts having pure tin plating on the leads.

Specific IC packages have definite failures associated with their use. These include

- *Ball grid array* (BGA) or *Overmolded pad-array carrier* (OMPAC): In manufacture, the primary failure mode for this package is short circuits (98% vs. 2% open circuits).
- *Flip chip:* In manufacture, the primary failure mode for a flip chip on an MCM module is a short circuit.
- PQFP: Packages are susceptible to moisture-induced cracking in applications requiring reflow soldering.
- Quad flatpack (QFP): The most common quad flatpack manufacturing failure mode is a short circuit (82% short circuits, 2% open circuits, and 16% misalignment problems).
- Tape carrier package (TCP): Tape automated bonding (TAB) package has an open circuit (98% vs. 2% misalignment problems) failure mode.

Table 15.1 uses the environmental conditions as screening tests to detect parts that may prematurely fail because of a problem or manufacturing defect. The table shows the various screening tests that can be used to detect the various defects that effect microcircuit reliability. The screens that are rated as good to excellent in identifying component defects, and that are low in cost, are high-temperature storage, temperature cycling, thermal shock, nitrogen bomb test (this tests hermetic package seals), and gross leak test (which also tests hermetic package seals). Screening tests are done prior to putting devices in stock for future use. Proper ESD procedures and controls should be followed, and because of ESD degradation effects the parts should not be periodically handled or tested; only electrically test devices prior to use.

Component failures, such as pin to pin leakage, can also be caused by contaminating films that are on the seal and embedment glass of the package. One such failure mode reported for SRAMs was caused by lead sulfide from antistatic ABS/PVC (acrylonitrile-butadiene-styrene/polyvinyl chloride) plastic trays used to store parts during assembly and postseal operations. The lead sulfide exhibited itself as a shiny film over the seal and embedment glass and resulted in more than 20 μA (at 5 V) leakage. PVC plastic that uses

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Thioglycolate (or Dibutyltin bis or Iso Octylthioglicolate, heat stabilizers), a common additive, can cause sulfur-based contamination (and corrosion) on parts with lead-based embedment glass. Because of this, PVC manufacturers are pursuing alternate stabilizer materials, and products that can have performance degradations are often stored and processed using metal trays.

15.6 Lead Finish

Reducing lead in the environment is an environmental health and safety issue that is of prime importance today. While a typical microprocessor has approximately 0.2 g of lead and a computer motherboard has 2 to 3 g of lead, and the whole USA electronic interconnection market uses less than 2% of the world's lead production, to address environmental concerns, there are various environmental directives limiting the use of lead and other hazardous substances. These directives, formulated by the European Union in a Directive of the European parliament and of the Council of the European Union are:

- End of Life Vehicles (ELV) Directive 2000/53/EC in force since October 21, 2000, requires that products be free of heavy metals such as mercury, cadmium, hexavalent chromium and lead. It requires a recycling system be established by vehicle manufacturers for vehicles made after July 1, 2002 and for all vehicles regardless of date of manufacture by July 1, 2007. Lead can still be used as an alloying additive in copper and in solderable applications.
- The WEEE Directive, Waste Electrical and Electronic Equipment (WEEE) Directive 2002/96/EC, expands the recycling requirements of the ELV Directive to include a broad range of electronic and electrical appliances and equipment. WEEE went into effect on February 13, 2003. It is to be scheduled to become European national law by August 13, 2004, and be applicable to consumer use products by August 13, 2005. Article 2(3) however states "Equipment which is connected with the protection of the essential interests of the security of Member States, arms, munitions and war material shall be excluded from this Directive. This does not, however, apply to products which are not intended for specifically military purposes."
- RoHS Directive, The Restriction of Hazardous Substances in Electrical and Electronic Equipment, this Directive 2002/95/EC establishes standards and limits for the hazardous material content in electronic and electrical equipment. The Directive went into effect on February 13, 2003. It is scheduled to become European national law by August 13, 2004 for be in force for products by July 1, 2006. Banned or restricted substances include lead, mercury, cadmium, hexavalent chromium, certain brominated flame retardants (PBBs), and polybrominated diphenyl ethers (PBDEs).

The recommended lead-free solder formulation is Sn-Ag-Cu for board assembly but there are other formulations such as Nickel-Palladium (NiPd), or Nickel-Palladium with Gold flash (NiPdAu). Passive components, to be compatible with a lower temperature Lead process (which is 215°C for 50/50 Tin/Lead formulations and 230°C for 40/60 formulations) and the higher lead-free process of up to 260°C, use pure matte Tin for their contacts. The use of lead in solder is partially based on several potential reliability issues. Pure Tin component leads have been shown to result in inter-metallic migration in the termination of the electronic component and the growth of tin whiskers which could cause short circuits (which is why there is a exemption for military use (only) components).

The National Electronics Manufacturing Initiative (NEMI) has addressed the problem of "tin whiskers" in lead-free assemblies. A tin whisker is defined by them as

A spontaneous columnar or cylindrical filament, which rarely branches, of monocrystalline tin emanating from the surface of a plating finish. Furthermore, tin whiskers may have the following characteristics:

- an aspect ratio (length/width) greater than 2
- can be kinked, bent, twisted
- generally have consistent cross-sectional shape
- may have striations or rings around it

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Their recommended test method is

- temperature cycling (−55°C to + 85°C, approximately 3 cycles/hour)
- temperature humidity tests at 60°C/93% RH
- ambient storage (air-conditioned facility)

(Note: Tin will undergo a phase transformation, becoming a powdery form (called Tin Pest) if Tin plated parts are stored for more than 1 week at temperatures below 13°C).

15.7 Screening and Rescreening Tests

In the 1970s, the Navy instituted a policy requiring devices be rescreened on receipt (by the contractor or an independent test laboratory) because there was evidence that Joint Army Navy (JAN) qualified devices (including JANTX, JANTXV discrete component) were not of a quality level to be used in military hardware (DoD directive 4245.7-M Transition from Development to Production, which was signed in 1984 and saw wide implementation by the end of 1986).

Rescreening tests often imposed included the following:

Destructive physical analysis (DPA) examination tests, where two pieces of each lot of parts (as a minimum) were cut open and examined to determine the workmanship of the parts. If the workmanship was judged to be poor, the whole lot of parts were rejected (a lot of parts is defined as parts from one manufacturer from one assembly line and one date code).

Particle impact noise detection (PIND) tests, where each hybrid part or IC with a cavity where the die was unglassivated (insulated with a glass coating, Fig. 15.5 shows an IC with cracks in this coating), was vibrated and transducers, mounted on the part would detect if there were any particles rattling around. Parts with loose pieces in them were rejected from the shipment.

Go-no go electrical tests, and static and dynamic tests were required to be performed at low, ambient (25°C) and high temperatures.

Hermeticity testing was required to test the integrity of the package seal.

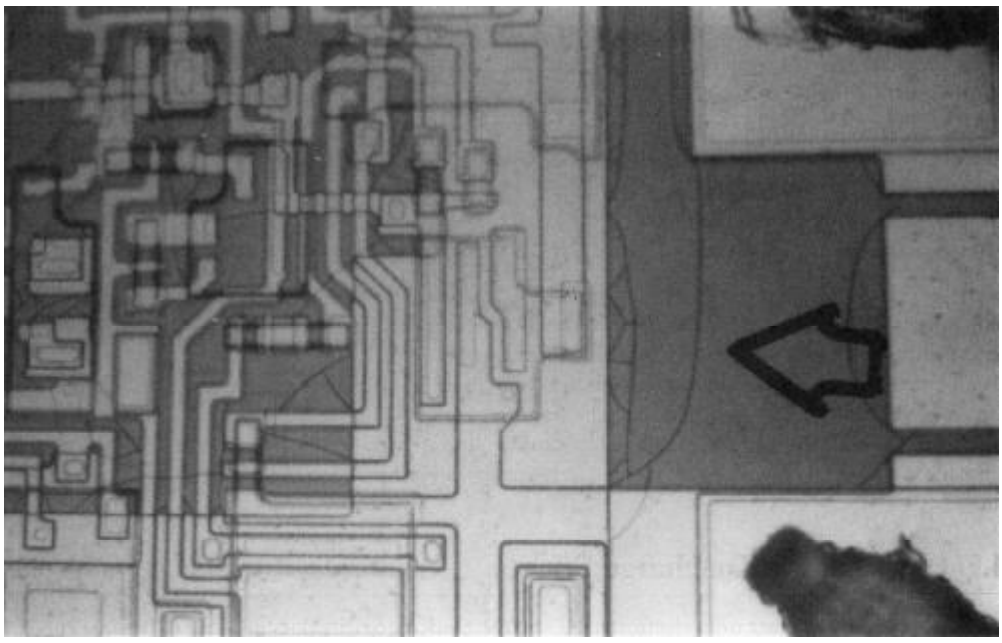


FIGURE 15.5 An IC with cracks in the die glass coating.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

To review the reasons for component rescreening we need to examine test data for the time period involved. In 1981 it was reported that the Naval Weapons Support Center (NWSC) in Crane, Indiana, found part defect rates of 15% for ICs and 17% for discrete semiconductors. In a January 1983 report, one independent test laboratory found reject rates of 16.7% for linear ICs, 8.4% for digital ICs, and 9.2% for CMOS ICs. By October 1984, this same laboratory found defect rates of 5.5% for linear ICs, 3.7% for digital ICs, 8.2% for transistors, and 3.8% for optoelectronics. In 1983, the Defense Electronic Center found semiconductor defect rates of 8% for nonmilitary specification parts, with a less than 1% defect rate for military specification devices. In 1986, a military contractor estimated rescreening failures of 0.9% for ICs and 1.5% for transistors. In 1989, a manufacturer of a special purpose computer workstation for the U.S. Navy reported the following rescreening results: IC/semiconductors tested: 8312 (127 types of parts); total rejects PIND/DPA: 25 (16 PIND failures on hybrid components, 6 electrical failures, 3 mechanical failures—broken leads); reject rate: 0.30%.

Breaking down the hybrid failures, of 99 military qualified oscillators tested, 6 were rejected for a rejection rate of 6.1%; 53 nonmilitary qualified oscillators were tested of which 10 were rejected, for a rejection rate of 18.8%.

The Semiconductor Industry of America conducted a quality statistics program to monitor and report on industry data on various quality control indices and parameters of microcircuits from 1985 to the early 1990s. In 1991, the analysis of the data showed that for every 10,000 parts shipped, there is on average only one part with electrical defects, or a 100 **parts per million (PPM)** defect rate. The data were reported using JEDEC standard 16, Assessment of Microcircuit Outgoing Quality Levels in Parts Per Million and represents data on JAN qualified parts, DESC drawing parts, MIL-STD-883C screened devices, and devices procured to source control drawings (SCDs). Over 150 million devices were sampled with the reporting companies supplying about 80% of all military microcircuits.

The breakdown of defects based upon part type are linear parts: 100 PPM, bipolar digital 50 PPM (10 PPM in the first quarter of 1991), MOS digital: <100 PPM, and MOS memory: 160 PPM (with defects falling an average 30 PPM/quarter). If all the data are combined, the data indicate that for every 25,000 parts shipped, there averages only 1 part with electrical defects. The study also shows that there is a steady and continual decline in the level of electrically defective parts. This analysis shows a significant improvement in part quality since the 1970s, when component rescreening was instituted, and rescreening is probably not necessary except for very high-reliability applications (such as space applications).

Semiconductor, Integrated Circuit, and Hybrid Device Screening Tests

Table 15.1 is a summary of the various screening tests that can be used to detect semiconductor failure mechanisms. Failure mechanisms independent of application environment (oxide or diffusion defects) can be accelerated by operating the device, i.e., performance of a **burn-in** test.

Failure mechanisms dependent on the application environment (bond or metallization defects) can be accelerated by temperature or mechanical stresses (vibration).

Time and environment dependent failure mechanisms (metal migration, corrosion, intermetallic compound formulations caused by dissimilar use) can be accelerated by device operation (i.e., elevated temperature burn-in testing).

15.8 Electrostatic Discharge Effects

ICs are susceptible to damage from electrostatic discharge. Static electricity does not have to be large enough to cause a visible spark for an IC to be damaged. Figure 15.6 is an ESD failure in a D/A converter circuit. Figure 15.7 shows a passivation fault extending under the metallization.

To prevent ESD damage, various materials are used. Topical antistats, such as those that may be used to coat some plastic IC dip shipping tubes, include detergent substances, which wet the surface being treated. These coatings depend on their moisture content and work most effectively at high humidities. However, these coatings do wear out and treated surfaces must be checked periodically with a static detector.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

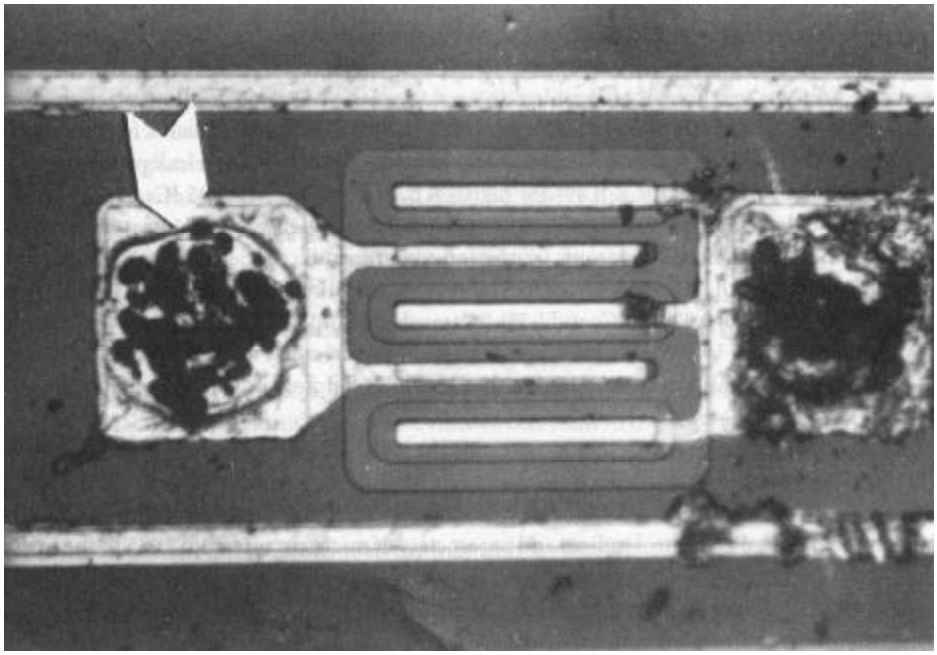


FIGURE 15.6 An ESD failure in a D/A converter circuit.

FIGURE 15.7 Passivation fault extending under the metallization.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Treated IC dip tubes should be retreated after use as the IC leads can scrape away the coating as they slide out of the tube.

Pink poly bags are also used to contain components or assemblies. These bags also lose their effectiveness, because they rely on absorbed moisture within the material, and must be checked periodically.

Nickel coated bags may have conductivity discontinuities due to the thin coating cracking after repeated bag handling. These bags should be periodically tested and examined for physical damage.

Defining Terms

Alpha particle: A product given off by the decay of radioactive material (usually emitted by traces of radioactivity in IC ceramic packaging materials). This particle has a positive charge equal to twice that of an electron and is emitted at a very high velocity. Alpha particles can cause temporary memory upsets in DRAMs, which is known as *soft error rate*.

Bond pad: Areas of metallization on the IC die that permit the connection of fine wires or circuit elements to the die. (See also wire bond.)

Burn-in: Component testing where infant mortality failures (defective or weak parts) are screened out by testing at elevated voltages and temperatures for a specified length of time.

Ceramic: An inorganic, nonmetallic clay or glasslike material whose final characteristics are produced by subjection to high temperatures.

DC test: Tests that measure a static parameter, for example, leakage current.

Defense Electric Supply Center (DESC) part: Used to denote a Standard Military drawing part that has been approved by the DESC, in Columbus (formerly in Dayton), Ohio.

Die: A single IC, also known as a chip.

Die information exchange (DIE): A specification created by chip makers and software vendors to provide basic die information in a standard format.

DIE bonding: The attachment of an IC chip (or DIE) to a substrate or a header.

DIE separation: Refers to the separation of the actual microcircuit chip from the inside of the package.

Destructive physical analysis (DPA): Devices are opened and analyzed for process integrity and workmanship.

Electrostatic discharge (ESD): The instantaneous transfer of charges accumulated on a nonconductor to a conductor, into ground.

Failure in time (FIT): A rating equal to the number of failures in one billion (10^9) h.

Frit: A relatively low softening point material of glass composition.

Function test: A check for correct device operation generally by truth table verification.

Hermetic: Sealed so that the object is gas tight (usually to a rate of less than 1×10^{-6} cc/s of helium.)

Joint Army Navy (JAN): When used when referring to microcircuits indicated a part fully qualified to the requirements of MIL-M-38510 for ICs (now replaced by MIL-M-38535) and MIL-S-19500 for semiconductors. JAN class B microcircuit level of the standard military drawing (SMD) program is the preferred level for design in new weapons systems.

JANTX: A prefix denoting that the military specification device has received extra screening and testing, such as an 100% 168-h burn-in.

JANTXV: A JANTX part with an added precapsulation visual requirement.

Joint Electron Device Engineering Council (JEDEC): A part of the EIA.

Lead: A conductive path, usually self-supporting, the portion of an electrical component that connects it to outside circuitry.

Lead frame: The metallic portion of the device package that makes electrical connections from the die to other circuitry.

Mask: The stencil of circuit elements through which light is shown to expose that circuit pattern onto a photoresist coating on the chip (die). The exposed areas are stripped away leaving a pattern.

Metallization: The deposited thin metallic coating layer on a microcircuit or semiconductor.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Passivation: The process in which an insulating dielectric layer is formed over the surface of the die. Passivation is normally achieved by thermal oxidation of the silicon and a thin layer of silicon dioxide is obtained in this manner (a combination of PECVD oxide and PECVD nitride deposited at lower temperature (below 450°C)). Other passivation dielectric coatings may also be applied (used), such as silicon glass (silicon oxynitride).

Particle impact noise detection testing (PIND): A test where cavity devices are vibrated and monitored for the presence of loose particles inside the device package via the noise the material makes. These loose particles may be conductive (such as gold flake particles from gold eutectic die attachment operations) and could result in short circuits. This is not required on a 100% basis for military class B devices. It is required for all class S, or devices generally required for spacecraft application use.

Popcorning: A plastic package crack or delamination that is caused by the phase change and expansion of internally condensed moisture in the package during reflow soldering, which results in stress the plastic package can not withstand.

Parts per million (PPM): The number of failures in one million parts. A statistical estimation of the number of defective devices, usually calculated at a 90% confidence level.

Schmoo plot: An X–Y plot giving the pass/fail region for a specific test while varying the parameters in the X and Y coordinates.

Soft error: An error, or upset in the output of a part (usually applies to memory devices for a single bit output error), which does not reoccur (i.e., the device performs to specifications when tested after the failure occurred).

Substrate: The supporting material upon which the microcircuit (IC) is fabricated, or in hybrids the part to which the IC (and other components, etc.) is attached.

Tin whisker: A hairlike single crystal growth formed on the metallization surface.

Wire bond: A wire connection between the semiconductor die bond pad and the leadframe or terminal.

Acknowledgement

I would like to thank Ron Kalakuntla of ISSI (Integrated Silicon Solution Inc.) and Mr. Lawrence Taccour, of Technical Marketing Group, for their assistance in reviewing this chapter.

References

Arnold, H.D. 1981. Institute of Environmental Sciences Proceedings. ESSEH (Environmental Stress Screening of Electronic Hardware), Institute of Environmental Sciences (Mt. Prospect, IL) Conf., Sept. 21.

Bley, W. 1981. Testing high-speed bipolar memories. Fairchild Camera and Instrument Corp., Mountain View, CA, Nov.

CALCE (Computer Aided Life Cycle Engineering) News, Jan. 1993. Workshop On Temperature Effects, University of Maryland, College Park, MD.

Chester, M. 1986. DOD's rescreening of chips stirs controversy. *Electronic Products,* Oct. 15, pp. 83–84.

Costlow, T. 1995. MCM substrates mixed. *Electronic Engineering Times,* Jan. 16.

Ellis, M. 1984. Non-Mil Defects Surpass Mil. *Electronic Buyers' News,* Nov. 26, p. 6.

Frye, M.A. 1994. Extension of the implementation date of MIL-I-38535 and MIL-S-19500 regarding the prohibition of pure tin as a plating material. Defense Logistics Agency, Defense Electronics Supply Center, Dayton, OH (Letter dated Jan. 21).

GIDEP. Government and Industry Data Exchange Program (Corona, CA), Problem Advisory, 7G-P-95-01 (Jan. 1995), Microcircuits, Flux, Soldering, Liquid; Problem Advisory,G4-P-93-01, Packaging, Cardboard, Flatpack, Microcircuit; Problem Advisory, ZW-P- 9301A (Jan. 1993), Storage Container, Material, Contamination; Problem Advisory, S4-P-93-01 (Jan. 1993), Transistor, Tin Plating, Whisker Growth.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Gulley, D.W. 1992. Texas Instruments, mean time between events, a discussion of device failures in DRAMs. 9(5), Sept.
Hamilton, H.E. 1984. Electronics test. Micro Control Co. Minneapolis, MN, April.
Hitachi. 1991. Reliability report, Multiport Video RAM. HM534251, MC-883, Hitachi, Ltd., Tokyo, Japan, Oct. 14.
Hnatek, E.R. 1983. The case for component rescreening. *Test & Measurement World,* Jan., pp. 18–24.
Hnatek, E.R. 1984. ICs for military and aerospace show dramatic jump in quality, reliability, military/space electronics design. Viking Labs., Inc. *Military/Space Electronics Design* (McGraw-Hill), Oct., pp. 27–30.
Hu, J.M., Barker, D., Dasgupta, A., and Arora, A. 1993. Role of failure-mechanism identification in accelerated testing. *Journal of the IES* (July/Aug.), Institute of Environmental Sciences (Mt. Prospect, IL).
ESSEH (Environmental Stress Screening of Electronic Hardware) 1985. Environmental stress screening for parts. Institute of Environmental Sciences Proceedings, *ESSEH*, Sept.
Klinger, D.J., Nakada, Y., and Memendez, M.A. *AT&T Reliability Manual,* Van Nostrand Reinhold,1990.
Lawrence, J.D. Jr. 1983. *Parallel testing of memory devices.* Reliability Inc., Houston, TX, Oct.
RADC. 1988. *Reliability/maintainability/testability design for dormancy*. Lockheed Electronics Co., Rome Air Development Center, Rept. RADC-TR-88-110, May. (Available from the Defense Technical Information Center (document #AD-A202-704)), Defense Logistics Agency, Department of Defense.
Meeldijk, V. 1995. *Electronic Components: Selection and Application Guidelines.* Wiley Interscience, New York, Chap. 10 and 11.
Meeldijk, V. 1990. Effects of storage and dormancy on components. *Electronic Servicing and Technology Magazine,* Dec., pp. 6–11.
Micron Technology, Inc., Boise, ID. 1991. *Quality/Reliability Handbook,* 4/91, Reliability Monitor, 4 M DRAM, rev. 10/91 and 1 MEG DRAM book.
MIL-HDBK. 1988. *Electronic Reliability Design Handbook.* MIL-HDBK-338 Military Handbook, Oct. 12.
MIL-STD. 1995. NASA standard electrical, electronic, and electromechanical (EEE) parts list. MIL-STD-975, March 17.
MIL-STD. 1992. Electronic parts, materials, and processes for space and launch vehicles. MIL-STD-1547, Dec. 1.
MIL-STD-19500. 1994. General specification for semiconductor devices, April 15.
Motorola, Phoenix, AZ. 1982. Dynamic RAM quality and reliability report. MCM6664A/6665A 64K.
Murray, J. 1994. MCMs pose many production problems. *Electronic Engineering Times,* June 11.
Navsea Systems Command. 1991. Washington, D.C. *Parts application and reliability information manual for Navy electronic equipment*. TE000-AB-GTP-010, Navsea Systems Command (stock number 0910-LP-494-5300), March.
O'Connor, P.T.D. 1985. *Practical Reliability Engineering,* 2nd ed. Wiley, New York.
RAC. 1991. *Failure mode/mechanism distributions*. FMD-91, Reliability Analysis Center, Rome, NY.
School, R. 1985. *Effective screening techniques for dynamic RAMs*. Pacific Reliability Corp. presented at Electro., Institute of Electrical and Electronic Engineers.
Schaefer, S. 1994. DRAM soft error rate calculations. *Design Line,* 3(1).
Semiconductor Industry Association (San Jose, CA). 1991. *Quality statistics report on military integrated circuits*. Government Procurement Committee.
Smith, D.J. 1985. *Reliability and Maintainability in Perspective,* 2nd ed. Halsted Press, Wiley, New York.
Somos, I.L., Eriksson, L.O., and Tobin, W.H. 1986. Understanding di/dt ratings and life expectancy for thyristors. *PCIM Magazine*, Feb.
U.S. Army Material Command AMCP-706-196.
Willoughby, W.J. Jr. 1980. Military electronics/countermeasures: View from the top (interview), Aug., pp. 14–20, 60–61.
Yalamanchili, P., Gannamani, R., Munamarty, R., McClusky, P., and Christou, A. 1995. Optimum processing prevents PQFP popcorning. CALCE Electronic Packaging Research Center, MD, *SMT*, May.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

Further Information

The following sources can be referenced for additional data on failure modes:

The Reliability Analysis Center (RAC), Rome, NY, Failure Mode/Mechanism Distributions, 1991, FMD-91.

AT&T Reliability Manual, by David J. Klinger, Yoshinao Nakada, and Maria A. Memendez, published by Van Nostrand Reinhold, 1990.

© 2006 by Taylor & Francis Group, LLC

PDF created with pdfFactory Pro trial version www.pdffactory.com

16

Fundamental Computer Architecture

16.1 Introduction **16**-1
16.2 Defining a Computer Architecture **16**-1
16.3 Single Processor Systems **16**-2
16.4 Multiple Processor Systems **16**-6
16.5 Memory Hierarchy **16**-7
16.6 Implementation Considerations **16**-8
Packaging Considerations • Technology Considerations • Wafer Scale Integration (WSI) • Multichip Modules (MCMs)

Joy S. Shetler

16.1 Introduction

The design space for computer **architectures** is fairly diverse and complicated. Each new architecture strives to fulfill a different set of goals and to carve out a niche in the computer world. A system can be composed of one or several processors. Many concepts apply to both multiple processor and single processor systems. Many researchers have concluded that further advances in computational speed and throughput will come from parallelism rather than relying heavily on technological innovation, as has occurred in the past. But implementation is still an important consideration for any computer system.

16.2 Defining a Computer Architecture

Some important attributes of a computer system are as follows:

- Structure of the control path(s)
- Structure of the data path(s)
- The memory organization
- The technology used in the implementation
- The number of clocks, that is, single clocked or dual clocked
- The clock speed(s)
- The degree of pipelining of the units
- The basic topology of the system
- The degree of parallelism within the control, data paths, and interconnection networks

© 2006 by Taylor & Francis Group, LLC

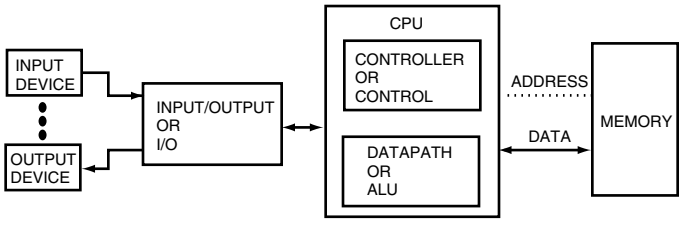


FIGURE 16.1 Block diagram of a single processor system.

In some cases, the **algorithms** used to execute an instruction are important as in the case of data flow architectures or systolic arrays. Some of these attributes are dependent on the other attributes and can change depending on the implementation of a system. For example, the relative clock speed may be dependent on the technology and the degree of **pipelining**. As more diverse and complicated systems are developed or proposed, the classification of a computer architecture becomes more difficult. The most often cited classification scheme, Flynn's taxonomy (Flynn, 1966) defined four distinct classes of computer architectures: *single instruction, single data* (SISD), *single instruction, multiple data* (SIMD), *multiple instruction, single data* (MISD), and *multiple instruction, multiple data*, (MIMD). A particular classification is distinguished by whether single or multiple paths exist in either the control or data paths. The benefits of this scheme are that it is simple and easy to apply. Its prevalent use to define computer architectures attests to its usefulness. Other classification schemes have been proposed, most of which define the parallelism incorporated in newer computer architectures in greater detail.

16.3 Single Processor Systems

A single processor system, depicted in Fig. 16.1 is usually composed of a controller, a data path, a memory, and an input/output unit. The functions of these units can be combined, and there are many names for each unit. The combined controller and data path are sometimes called the central processing unit (CPU). The data path is also called an arithmetic and logical unit (ALU).

Hardware parallelism can be incorporated within the processors through pipelining and the use of multiple functional units, as shown in Fig. 16.2. These methods allow instruction execution to be overlapped in time. Several instructions from the same instruction stream or *thread* could be executing in each pipeline stage for a pipelined system or in separate functional units for a system with multiple functional units. **Context** is the information needed to execute a process within a processor and can include the contents of a program counter or instruction counter, stack pointers, instruction registers, data registers, or general

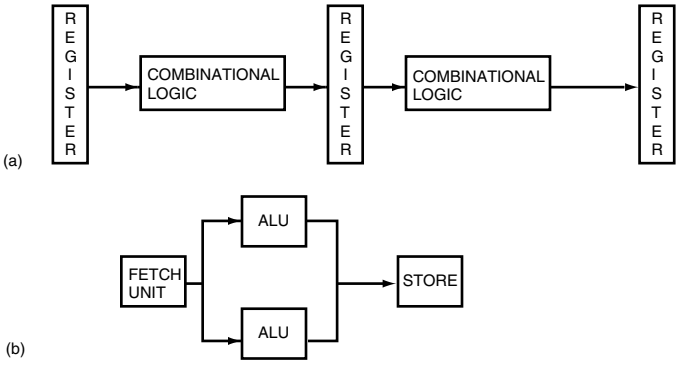


FIGURE 16.2 Methods of incorporating parallelism within a processor system: (a) pipelines, (b) multiple units.

© 2006 by Taylor & Francis Group, LLC

purpose registers. The instruction, data, and general purpose registers are also sometimes called the register set. Context must be stored in the processor to minimize execution time. Some systems allow multiple instructions from a single stream to be active at the same time. Allowing multiple instructions per stream to be active simultaneously requires considerable hardware overhead to schedule and monitor the separate instructions from each stream.

Pipelined systems came into vogue during the 1970s. In a pipelined system, the operations that an instruction performs are broken up into stages, as shown in Fig. 16.3(a). Several instructions execute simultaneously in the pipeline, each in a different stage. If the total delay through the pipeline is D and there are n stages in the pipeline, then the minimum clock period would be D/n and, optimally, a new instruction would be completed every clock. A deeper pipeline would have a higher value of n and thus a faster clock cycle.

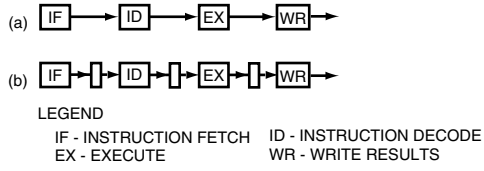


FIGURE 16.3 Comparison of the basic processor pipeline technique: (a) not pipelined, (b) pipelined.

Today most commercial computers use pipelining to increase performance. Significant research has gone into minimizing the clock cycle in a pipeline, determining the problems associated with pipelining an instruction stream, and trying to overcome these problems through techniques such as prefetching instructions and data, compiler techniques, and caching of data and/or instructions. The delay through each stage of a pipeline is also determined by the complexity of the logic in each stage of the pipeline. In many cases, the actual pipeline delay is much larger than the optimal value, D/n, of the logic in each stage of the pipeline. Queues can be added between pipeline stages to absorb any differences in execution time through the combinational logic or propagation delay between chips (Fig. 16.4). Asynchronous techniques including handshaking are sometimes used between the pipeline stages to transfer data between logic or chips running on different clocks.

It is generally accepted that, for computer hardware design, simpler is usually better. Systems that minimize the number of logic functions are easier to design, test, and debug, as well as less power consuming and faster (working at higher clock rates). There are two important architectures that utilize this concept most effectively, **reduced instruction set computers** (**RISC**) and SIMD machines. RISC architectures are used to tradeoff increased code length and fetching overhead for faster clock cycles and less instruction set complexity. SIMD architectures, described in the section on multiple processors, use single instruction streams to manipulate very large data sets on thousands of simple processors working in parallel.

RISC processors made an entrance during the early 1980s and continue to dominate small processor designs as of this writing. The performance p of a computer can be measured by the relationship

$$p = \left(\frac{\text{computations}}{\text{instruction}}\right)\left(\frac{\text{instructions}}{\text{cycle}}\right)\left(\frac{\text{cycles}}{\text{second}}\right)$$

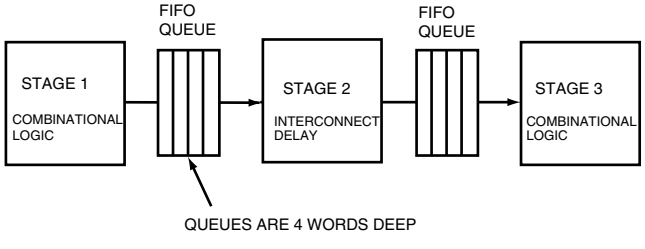


FIGURE 16.4 A pipeline system with queues.

© 2006 by Taylor & Francis Group, LLC

The first component (computations/instruction) measures the complexity of the instructions being executed and varies according to the structure of the processor and the types of instructions currently being executed. The inverse of the component (instructions/cycle) is commonly quoted for single processor designs as **cycles per instruction** (**CPI**). The inverse of the final component (cycles/second) can also be expressed as the clock period of the processor. In a RISC processor, only hardware for the most common operations is provided, reducing the number of computations per instruction and eliminating complex instructions. At compile time, several basic instructions are used to execute the same operations that had been performed by the complex instructions. Thus, a RISC processor will execute more instructions than a **complex instruction set computer** (**CISC**) processor. By simplifying the hardware, the clock cycle is reduced. If the delay associated with executing more instructions (RISC design) is less than the delay associated with an increased clock cycle for all instructions executed (CISC design), the total system performance is improved. Improved compiler design techniques and large on-chip caching has continued to contribute to higher performance RISC designs (Hennessy and Jouppi, 1991).

One reason that RISC architectures work better than traditional CISC machines is due to the use of large on-chip caches and register sets. Since locality of reference effects (described in the section on memory hierarchy) dominate most instruction and data reference behavior, the use of an on-chip cache and large register sets can reduce the number of instructions and data fetched per instruction execution. Most RISC machines use pipelining to overlap instruction execution, further reducing the clock period. Compiler techniques are used to exploit the natural parallelism inherent in sequentially executed programs.

A register window is a subset of registers used by a particular instruction. These registers are specified as inputs, outputs, or temporary registers to be used by that instruction. One set of instructions outputs become the next inputs in a register window for another instruction. This technique allows more efficient use of the registers and a greater degree of pipelining in some architectures.

Scaling these RISC concepts to large parallel processing systems poses many challenges. As larger, more complex problems are mapped into a RISC-based parallel processing system, communication and allocation of resources significantly affects the ability of the system to utilize its resources efficiently. Unless special routing chips are incorporated into the system, the processors may spend an inordinate amount of time handling requests for other processors or waiting for data and/or instructions. Using a large number of simple RISC processors means that cache accesses must be monitored (*snoopy caches*) or restricted (*directory-based caches*) to maintain data consistency across the system.

The types of problems that are difficult to execute using RISC architectures are those that do an inordinate amount of branching and those that use very large data sets. For these problems, the hit rate of large instruction or data caches may be very low. The overhead needed to fetch large numbers of new instructions or data from memory is significantly higher than the clock cycle, virtually starving the processor. Compiler techniques can only be used to a limited extent when manipulating large data sets.

Further increases in speed for a single stream, pipelined processor will probably come about from either increasing the pipeline depth, **superpipelining** or increasing the width of the data path or control path. The latter can be achieved by either issuing more than one instruction per cycle, **superscalar**, or by using a **very long instruction word** (**VLIW**) architecture in which many operations are performed in parallel by a single instruction. Some researchers have developed the idea of an orthogonal relationship between superscalar and superpipelined designs (Hennessy and Jouppi, 1991). In a superpipelined design, the pipeline depth is increased from the basic pipeline; whereas in a superscalar design, the horizontal width of the pipeline is increased (see Fig. 16.5).

To achieve an overall gain in performance, significant increases in speed due to superpipelining must be accompanied by highly utilized resources. Idle resources contribute little to performance while increasing overall system costs and power consumption. As pipeline depth increases, a single instruction stream cannot keep all of the pipeline stages in a processor fully utilized. Control and data dependencies within the instruction stream limit the number of instructions that can be active for a given instruction stream. *No operation* (NoOps) or null instructions are inserted into the pipeline, creating *bubbles*. Since a NoOp does not perform any useful work, processor cycles are wasted. Some strategies improve pipeline utilization using techniques such as prefetching a number of instructions or data, branch prediction, software pipelining,

© 2006 by Taylor & Francis Group, LLC

trace scheduling, alias analysis, and register renaming to keep the memory access overhead to a minimum. An undesirable consequence of this higher level of parallelism is that some prefetched instructions or data might not be used, causing the memory bandwidth to be inefficiently used to fetch useless information. In a single processor system this may be acceptable, but in a multiple processor system, such behavior can decrease the overall performance as the number of memory accesses increases.

Superscalar processors use multiple instruction issue logic to keep the processor busy. Essentially, two or three instructions are issued from a single stream on every clock cycle. This has the effect of widening the control path and part of the datapath in a processor. VLIW processors perform many operations in parallel using different types of functional units. Each instruction is composed of several operation fields and is very complex. The efficient use of these techniques depends on using compilers to partition the instructions in an instruction stream or building extra hardware to perform the partitioning at run time. Again both these techniques are limited by the amount of parallelism inherent in an individual instruction stream.

A solution to fully utilizing a pipeline is to use instructions from independent instruction streams or threads. (The execution of a piece of code specified by parallel constructs is called a thread.) Some machines allow multiple threads per program. A thread can be viewed as a unit of work that is either defined by the programmer or by a parallel compiler. During execution, a thread may spawn or create other threads as required by the parallel execution of a piece of code. Multithreading can mitigate the effects of long memory latencies in uniprocessor systems; the processor executes another thread while the memory system services cache misses for one thread. Multithreading can also be extended to multiprocessor systems, allowing the concurrent use of CPUs, network, and memory.

To get the most performance from **multithreaded** hardware, a compatible software environment is required. Developments in new computer languages and operating systems have provided these environments (Anderson, Lazowska, and Levy, 1989). Multithreaded architectures take advantage of these advances to obtain high-performance systems.

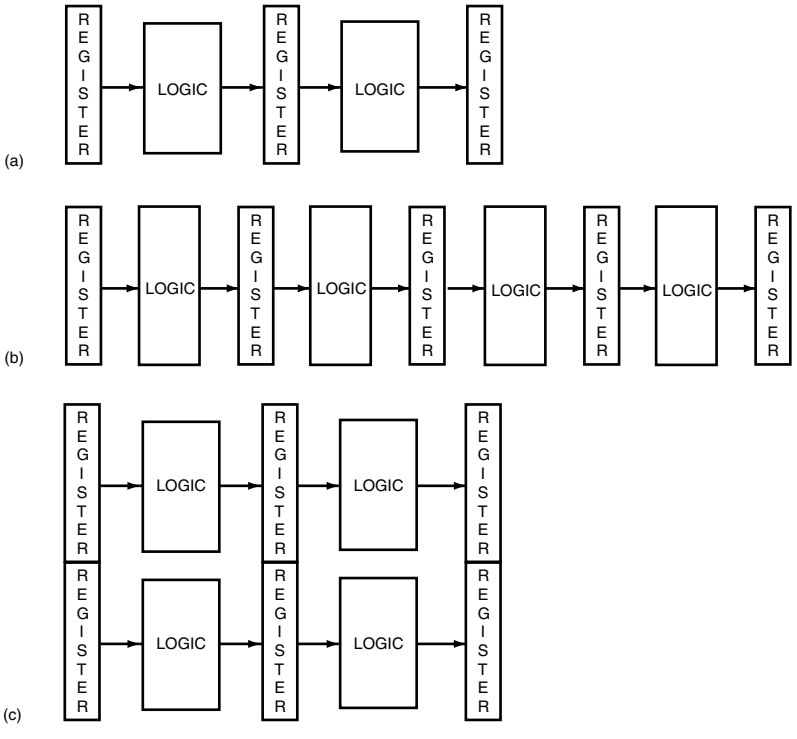


FIGURE 16.5 Superscalar and superpipelined systems: (a) pipelined, (b) superpipelined, (c) superscalar.

© 2006 by Taylor & Francis Group, LLC

16.4 Multiple Processor Systems

Parallelism can be introduced into a computer system at several levels. Probably the simplest level is to have multiple processors in the system. If parallelism is to be incorporated at the processor level, usually one of the following structures is used: SIMD, MIMD, or multicomputers.

A SIMD structure allows several data streams to be acted on by the same instruction stream, as shown in Fig. 16.6(a). Some problems map well into a SIMD architecture, which uses a single instruction stream and avoids many of the pitfalls of coordinating multiple streams. Usually, this structure requires that the data be bit serial, and this structure is used extensively in applications such as computer graphics and image processing. The SIMD structure provides significant throughput for these problems. For many applications that require a single data stream to be manipulated by a single instruction stream, the SIMD structure works slower than the other structures because only one instruction stream is active at a time. To overcome this difficulty structures that could be classified as a combination SIMD/MIMD structure have been applied. In a SIMD system, one instruction stream may control thousands of data streams. Each operation is performed on all data streams simultaneously.

MIMD systems allow several processes to share a common set of processors and resources, as shown in Fig. 16.6(b). Multiple processors are joined together in a cooperating environment to execute programs. Typically, one process executes on each processor at a time. The difficulties with traditional MIMD architectures lie in fully utilizing the resources when instruction streams stall (due to data dependencies, control dependencies, synchronization problems, memory accesses, or I/O accesses) or in assigning new processes quickly once the current process has finished execution. An important problem with this structure is that processors may become idle due to improper load balancing. Implementing an operating system (OS) that can execute on the system without creating imbalances is important to maintain a high utilization of resources.

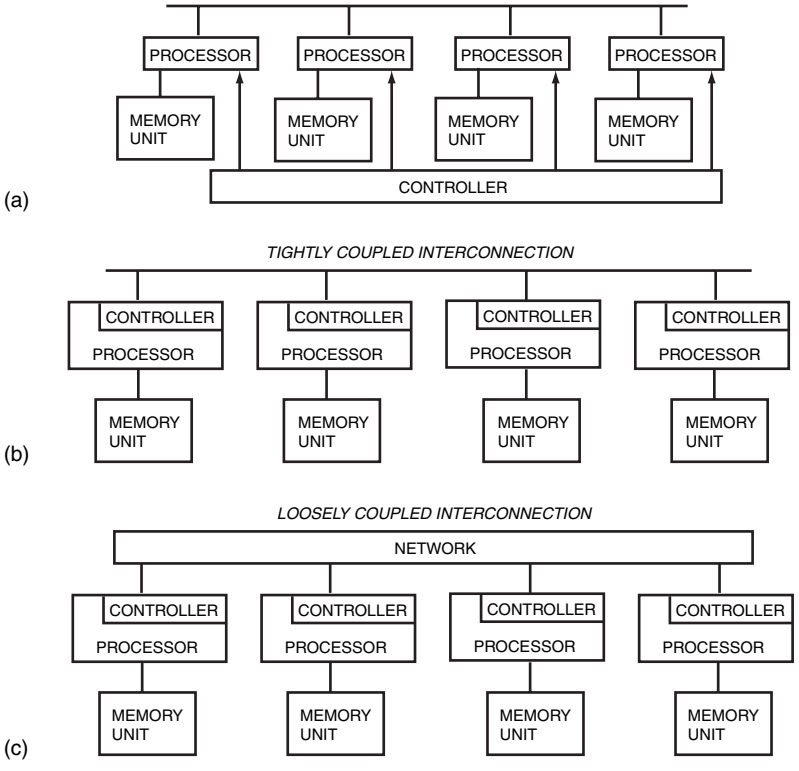


FIGURE 16.6 System level parallelism: (a) SIMD, (b) MIMD, (c) multicomputer.

© 2006 by Taylor & Francis Group, LLC

The next system, which is also popular due to its simple connectivity, is the distributed system or *multicomputer*. A network connects independent processors as shown in Fig. 16.6(c). Each processor is a separate entity, usually running an independent operating system process. A multicomputer will usually use message passing to exchange data and/or instruction streams between the processors. The main difficulties with the multicomputer are the latency involved in passing messages and the difficulty in mapping some algorithms to a distributed memory system.

16.5 Memory Hierarchy

High-performance computer systems use a multiple level memory hierarchy ranging from small, fast cache memory to larger, slower main memory to improve performance. Parallelism can be introduced into a system through the memory hierarchy as depicted in Fig. 16.7. A cache is a small, high-speed buffer used to temporarily hold those portions of memory that are currently in use in the processor.

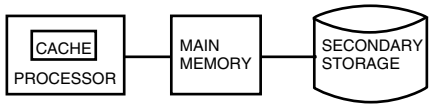


FIGURE 16.7 A common method of incorporating parallelism into a system.

Cache memories take advantage of the property of temporal and spatial locality in program behavior. Temporal locality states that if an item is referenced one point in time, it will tend to be referenced again soon. Spatial locality refers to the likelihood that if an item is referenced, nearby items will tend to be referenced soon.

A cache miss will cause the processor to request the needed data from the slower main memory. The average access time depends on the cache miss rate, the number of cycles needed to fetch the data from the cache, and the number of cycles required to fetch data from main memory in case of a cache miss.

Caches can be divided into separate instruction and data caches or unified caches that contain both instructions and data. The former type of cache is usually associated with a Harvard style architecture in which there are separate memory ports for instructions and data.

In most approaches to instruction fetch, the single chip processor is provided with a simple on-chip instruction cache. On a cache miss, a request is made to external memory for instructions and these off-chip requests compete with data references for access to external memory.

Memory latency is a major problem for high-performance architectures. The disparity in on-chip to off-chip access latencies has prompted the suggestion (Hwang, 1993) that there are four complementary approaches to latency hiding:

- Prefetching techniques
- Using coherent caches
- Using relaxed memory consistency models
- Using multiple contexts or multithreading within a processor

To maximize the performance benefit of multiple instruction issue, memory access time has to be kept low and cache bandwidth has to be increased. Unfortunately, accessing the cache usually becomes a critical timing path for pipelined operation.

Extending caching into a multithreaded processor creates some new problems in cache design. In single-threaded caches, on a context switch (when the processor begins execution of another thread), the cache is flushed and the instructions from the new thread are fetched. A multithreaded cache supports at least two different instruction streams, using different arrays for different threads. Another strategy is to allow the threads to share the cache, making it unnecessary to flush the cache on a context switch. The state of the old thread is stored in the processor, which has different register files for each thread. The cache sends the thread number and the instruction to the processor's instruction buffers. To keep the execution units supplied with instructions, the processor requires several instructions per cycle to be fetched into its instruction buffers. In this variable instruction fetch scheme, an instruction cache access may

© 2006 by Taylor & Francis Group, LLC

be misaligned with respect to a cache line. To allow line crossing, an alignment network can be used to select the appropriate instructions to be forwarded to the instruction buffers. Performance is increased by eliminating the need for multiplexing the data bus, thus reducing the I/O delay in a critical timing path and providing simultaneous reads and writes into the cache. Although this operation seems relatively simple, several variables must be considered when designing a multithreaded cache such as: address translation and mapping, variable instruction and data fetching, cache line crossing, request protocols and queuing, line and block replacement strategies, and multiple access ports.

By placing caches, which contain a subset of the main memory, within each processor, several processors could be using the same memory data. When a processor modifies its cache, all other caches within the system with that data are affected. This usually requires that a snooping mechanism be used to update or invalidate data in the caches. These mechanisms usually add considerable hardware to the system and restrict the implementation of the interconnection network to a bus type network so that cache updates can be monitored.

16.6 Implementation Considerations

Rather than beginning by building a system and testing the behavior of a breadboard model, the first step is to simulate the behavior using simulation models that approximate the processor structures. The hardware should be modeled at two levels.

High level, hierarchical models provide useful analysis of different types of memory structures within the processor. Flows through the models and utilization parameters can be determined for different structures. Low level, *register transfer level* (RTL) simulation of specific structures usually give a more accurate picture of the actual hardware's performance. These structures may include caches, register files, indexed instruction and data caches, memory controllers, and other elements. *Hardware description languages*, have been developed to represent a design at this level. After simulation and modeling of the design have been completed, the design can be implemented using various technologies.

Although significant technological advances have increased the circuit speed on integrated circuit (IC) chips, this increase in circuit speed is not projected to result in a similar increase in computer performance. The intrachip and interchip propagation delays are not improving at the same rate as the circuit speed. The latency imposed by the physical methods used to transfer the signals between faster logic has limited the computer performance increases that could be practically obtained from incorporating the logic into the designs.

$$\text{Delay}_{\text{circuit}} = \text{Delay}_{\text{interconnect}} + \text{Delay}_{\text{logic}}$$

An example of this phenomena is evident in pipelined systems, where the slowest pipe stage determines the clock rate of the entire system. This effect occurs between the parts of a hardware circuit. The relative speedup (obtained by making one part of the circuit faster) is determined by the slowest part of the circuit. Even if propagation delays through the fastest logic tended toward zero delay, the propagation delay through the interconnect would limit the speedup through the system or block of logic. So that the speedup for a system with zero logic delay compared to a system with some logic delay would be

$$\text{Speedup} = \frac{\text{Delay}_{\text{with logic}}}{\text{Delay}_{\text{without logic}}}$$

$$\text{Speedup} = \frac{\text{Delay}_{\text{interconnect}} + \text{Delay}_{\text{logic}}}{\text{Delay}_{\text{interconnect}}}$$

Rather than suggesting that further speedups due to technological innovation are impossible, these arguments show that new methods must be used that compensate for the latencies imposed by physical limitations. As demonstrated in Fig. 16.8, the time it takes for a signal to be transferred between chips might be several times the clock period on a chip implemented in a high-speed circuits. The delay between chips is not only affected by the length of the interconnecting wires between the chips but also by the

© 2006 by Taylor & Francis Group, LLC

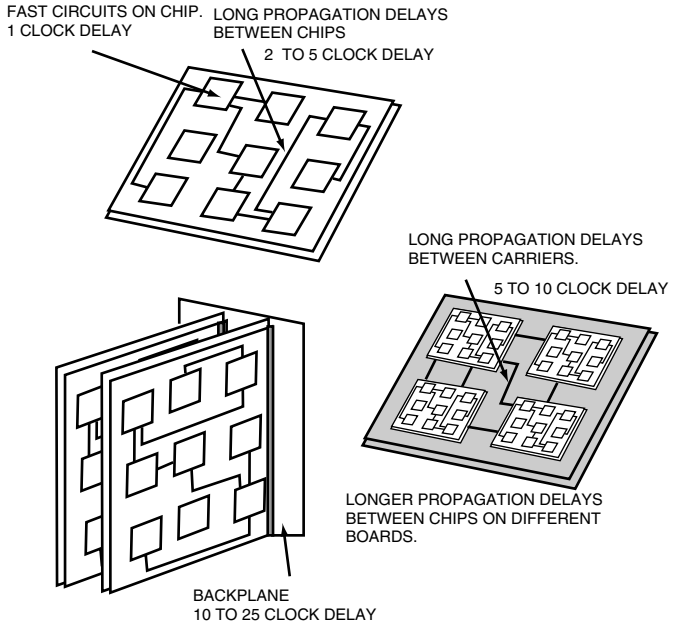


FIGURE 16.8 The relative propagation delays for different levels of packaging; propagation through interconnecting elements becomes longer relative to clock cycle on chip.

number and types of interconnections between the wires. Whenever signals must go through a via or pin to another type of material or package, mismatches in impedance and geometry can introduce reflections that affect the propagation delay through the path.

One way of synchronizing logic modules is to use self-timed circuits or **ternary logic** for the signal transfers. Ternary logic is compatible with some optical interconnection methods. Logic must be provided to translate the multiple valued signal back to a digital signal unless the entire system is designed in ternary logic. The overhead of designing the entire system in ternary logic is significant (50% increase in logic).

Pipelining the system so that one instruction is being transferred between chips while another instruction is executing is a promising technique to compensate for this latency. Pipelining techniques (Fig. 16.9) can be used to not only compensate for interconnect delay but also to monitor the inputs and outputs of a chip (when scan paths are designed into the latches), by inserting registers or latches at the inputs and outputs of the chips. This has the effect of increasing the length of the pipeline and the amount of logic, increasing the system latency, and increasing the layout area of the design. The main problems with this use of pipelining are clock synchronization and keeping the propagation delay through each pipeline stage symmetrical. Each register could be running on a different clock signal, each of which is *skewed* from the other, that is, one signal arrives at the pipeline register offset from the time that the other signal arrives at its pipeline register. Clock skew can cause erroneous values to be

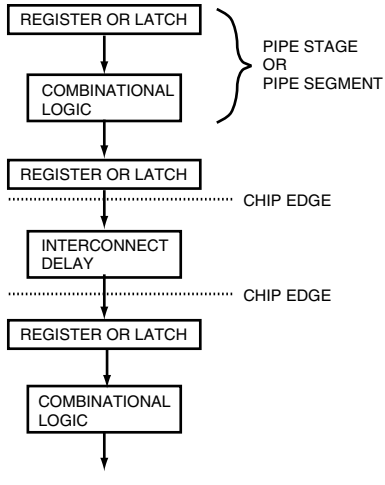


FIGURE 16.9 Using pipelining to compensate for interconnect delays.

© 2006 by Taylor & Francis Group, LLC

loaded into the next stage of the pipeline. For an optimal design, each pipeline stage would be symmetrical with respect to propagation delay. Any nonsymmetries result in longer clock periods to compensate for the worst-case propagation delay as mentioned previously. An ideal pipelined system must allow for clock skew and for differences in propagation delay through the stages of the pipeline.

Packaging Considerations

Ensuring high-bandwidth interconnections between components in a computer system is extremely important and some (Keyes, 1991) consider computer systems development to be the principal driving force behind interconnection technology. During early empirical studies of the pinout problem, it was found that in many designs there is a relationship between the amount of logic that can be implemented within a design partition and the number of I/O pins,

$$I = AB^r$$

where

I = the I/O pin count
B = the logic block count
A = the average size of the blocks
r = the *Rent exponent*

This relationship has popularly become known as Rent's rule based on the unpublished work by E.F. Rent at IBM. Although considerable research has been undertaken to disprove Rent's rule or to find architectures that overcome the pinout limitations, in many cases, there is a limitation on the amount of logic that can be associated with a given number of pins. This relationship severely affects the designs of some high-speed technologies where the pinout is fairly costly in terms of space and power consumption. Reduced pinout due to packaging constraints means that the logic must be kept simple and sparse for most designs.

RISC designs have benefited from the low number of pins and routing density available in VLSI chips. For a particular architecture certain relations between pinout and logic have been observed. Since the digital logic complexity available per interconnect is determined by empirical relationships, such as Rent's rule, reducing the complexity of a logic block (chip) will usually also reduce the number of interconnects (pins). Because of the simpler implementations, RISC chips have required less routing density and less pins than more complex designs.

Packaging techniques have been developed to provide increased pinout on IC chips. Most of these techniques have been developed for Si chip technology. These packaging techniques are beginning to spread to systems composed of other technologies. Most of the packaging techniques center around providing a higher density pinout by distributing pins across the chip instead of restricting the placement to the edges of the chip. Sometimes this involves using a *flip chip* technology, which turns the chip logic-side down on a board. Heat sinking—removing heat from the chip—becomes an issue as does being able to probe the logic.

Technology Considerations

Silicon semiconductor technology has been the most commonly used technology in computer systems during the past few decades. The types of Si chips available vary according to logic family such as n-channel metal-oxide semiconductor (NMOS), complementary metal-oxide semiconductor (CMOS), transistor transistor logic (TTL), and emitter coupled logic (ECL). Circuit densities and clock rates have continued to improve for these Si logic families, giving the computer designer increased performance without having to cope with entirely new design strategies. Higher speed technologies are preferred for high-end computer systems. Several technologies that hold promise for future computer systems include optical devices, gallium arsenide (GaAs) devices, superconductors, and quantum effect devices.

Many of the high-speed technologies are being developed using III–V materials. GaAs is the current favorite for most devices (sometimes used in combination with AlGaAs layers). Long and Butner (1990)

© 2006 by Taylor & Francis Group, LLC

provide a good introduction to the advantages and capabilities of GaAs circuits, including higher electron mobility (faster switching speeds), a semi-insulating substrate, and the major advantage of the suitability of GaAs semiconductors for optical systems. GaAs circuits are steadily increasing in circuit density to the VLSI level and becoming commercially available in large quantities. More designers are incorporating GaAs chips into speed-sensitive areas of their designs.

There are several GaAs circuit families. Some of the circuit families use only depletion mode field effect transistors (FETs), whereas others use both enhancement and depletion (E/D) mode FETs. Very large-scale integrated (VLSI) levels of integration are under development for *direct coupled FET logic* (DCFL) and *source coupled FET logic* (SCFL). By using enhancement and depletion mode FETs with no additional circuitry, DCFL is currently the fastest, simplest commercially available GaAs logic. SCFL (also using E/D FETs) provides high speed with greater drive capabilities but consumes more power and space than DCFL.

Several efforts are underway to harness photons for extremely fast interconnects and logic. Photons have a greater potential bandwidth than electrons, making optical interconnections attractive. Unlike electrons, photons have no charge and do not easily interact in most mediums. There are virtually no effects from crosstalk or electromagnetic fields on even very high densities of optical wires or signals. Although photon beams produce very high-quality signal transmission, the implementation of switching logic is difficult. The same immunity to interference that makes optics ideal for signal transmission makes switching logic difficult to implement. Most optical devices or systems under development exhibit high latencies in exchange for fast switching speeds. Optical interconnects, however, provide wider bandwidth interconnections for interchip or interprocessor communications. Optical interconnections have delays associated with converting signals from electrons to photons and back, increasing the propagation delay.

In a wire connection, the propagation characteristics limit the use of the wire to one pipeline stage. Reflected electrical signals must be dampened before a new signal can enter the wire, precluding the use of the wire by more than one digital signal during a time period. A wire interconnect would only act as a single stage of a pipeline, as shown in Fig. 16.9. An optical interconnect, due to its signal propagation characteristics, allows a finer degree of pipelining. Once a signal enters the optical interconnect, new signals can enter at regular time intervals without causing interference between the signals. Thus, an optical interconnect can represent several pipeline stages in a design, as shown in Fig. 16.10.

Two types of optical interconnects are under development, fiber optic and free space. Fiber-optic systems provide high bandwidth interconnections at speeds comparable to wire interconnects. Faster switching signals can be transported through fiber-optic systems longer distances than traditional wire and via interconnects. Since the fundamental physical limit for propagation delay between any two locations is the speed of light, free-space optical systems are, the ultimate choice for interconnection technology. The major drawback with optical interconnects is that the technologies that are currently available have a much lower density per area than traditional interconnects.

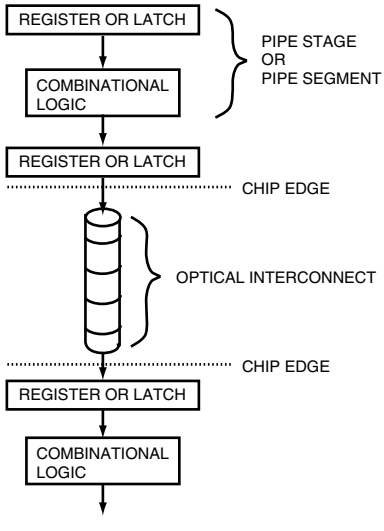


FIGURE 16.10 Using an optical connection in a pipeline system.

In a fiber-optic system, cables run between source and destination. The cables are usually wide (distances on the order of 250-μm centers between cables are common), difficult to attach, and subject to vibration problems. The lightguide material must maintain its low loss through all subsequent carrier processing (including the fabrication of electrical wiring and the chip-attachment soldering procedures) and the material used must be compatible with the carriers' temperature coefficient of expansion over the IC's

© 2006 by Taylor & Francis Group, LLC

operational temperature range. A benefit of using a fiber-optic cable interconnect is that a simple switching function using devices, such as directional optical couplers, can be performed in the cable.

In free-space systems, using just a detector and light source, signals can be transferred from one board to another in free space. The problem of connecting fibers to the boards is removed in a free-space system but the beam destination is limited to next nearest boards. Alignment of detector and sender becomes critical for small feature sizes necessary to provide high-density transfer of signals. For both free-space and fiber-optic systems, lower density or positional limitations makes optical interconnects unattractive for some computer architectures.

The near-term application of optical interconnects are in interprocessor communications with optical storage being the next leading application area. There are significant propagation delays through the translation circuitry when converting signals between the optical and the electrical realms. There are two leading materials for optical interconnect systems, GaAs and InGaAsP/InGaAs/InP. GaAs laser systems are by far the most developed and well understood. InGaAsP/InGaAs/InP *optoelectronic integrated circuits* (OEICs) are the next leading contender with current technology capable of providing laser functions for long-distance applications such as fiber-optic links for metropolitan area networks and CATV distribution.

Lowering the temperature of chips and interconnect results in an improvement in FET circuit switching speed and a reduction in the interconnect resistance. The reduction in resistance causes lower attenuation and greater bandwidth in the interconnect. These effects occur both in normal conductors and super conductors at low temperatures. Room temperature operation is usually preferred because of the problems with maintaining a low-temperature system and the extra power required to lower the temperature. Heat generated by devices on the chip can cause local *hot spots* that affect the operation of other circuits. The development of high-temperature superconductors increased the interest in using high bandwidth superconductor interconnect with high-speed circuits to obtain a significant increase in system clock rate.

Wafer Scale Integration (WSI)

One way to overcome the pinout problem is to reduce the pinout between chips by fabricating the chip set for a design on a single wafer. The logic partitions, which would have been on separate chips, are connected through high-speed metal interconnect on the wafer. Research involving **wafer scale integration** (**WSI**) dates back to some very early (1960s) work on providing acceptable yield for small Si chips. As processing techniques improved, circuit densities increased to acceptable levels, negating the need for WSI chips in the fledgling Si IC industry.

The driving force behind current WSI efforts is the need to provide high bandwidth communications between logic modules without suffering the degradation in performance due to off-chip interconnections. The problems with WSI are adequate yield, heat sinking, and pinout. The latter two problems can be coped with through using advanced packaging techniques such as flip-chip packages and water-cooled jackets. Obtaining adequate yields for mass production of wafers is probably the largest obstacle for using WSI systems. Processing defects can wipe out large sections of logic on wafers. Most WSI approaches use replicated or redundant logic that can be reconfigured either statically or dynamically to replace defective logic. This technique results in a working wafer if enough spares are available to replace the defective logic.

Architectures that are compatible with WSI are simple with easily replicated structures. Random access memory (RAM) chip designs have had the most successful use of WSI. The rationale for this approach is that GaAs is about at the stage of development that Si was in the early 1960s and, hence, WSI would provide a vehicle for a large gate count implementation of a processor. Large wafers are impractical due to the fragility of GaAs. Packaging techniques using defect-free GaAs chips mounted on another material might be more viable for some designs than WSI, due to the fragility of GaAs, the limited size of the GaAs wafers, and the cost of GaAs material.

In hybrid WSI, defect-free chips are mounted on a substrate and interconnected through normal processing techniques. [Note: the term hybrid is also used to refer to microwave components. A hybrid WSI component is also called a *wafer transmission module* (WTM).] This pseudo-WSI provides the density and performance of on-chip connections without having to use repair mechanisms for yield enhancement.

© 2006 by Taylor & Francis Group, LLC

The interconnections are the same feature sizes and defect levels as normal IC processing. Only a few planes are available to route signals, keeping the interconnect density low. Hybrid WSI has been suggested as a way for GaAs chips with optoelectronic components to be integrated with high logic density Si chips.

Multichip Modules (MCMs)

Multichip modules (MCMs) hold great promise for high-performance interconnection of chips, providing high pinout and high manufacturable levels of parts. MCMs are very close in structure to hybrid WSI. Active die are attached directly to a substrate primarily used for interchip wiring, introducing an extra level of packaging between the single die and the printed circuit board (PCB). The main difference between MCMs and hybrid WSI is the size of the substrate and interconnect. The MCM substrate can be as large as a wafer but usually is smaller with several thin-film layers of wiring (up to 35 or more possible).

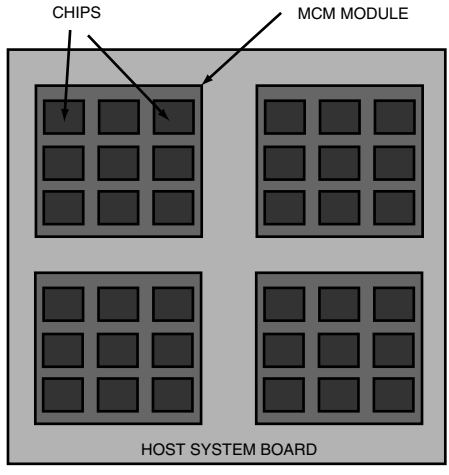


FIGURE 16.11 The basic MCM structure.

The feature size of the MCM interconnect is about 10 times that of the on-chip interconnect, reducing the interconnect disabled by processing defects to almost nil. By guaranteeing that only good die are attached to the substrates by burning-in the die under temperature and voltage, MCMs can be mass produced at acceptable yield levels. The basis MCM structure is shown in Fig. 16.11.

Defining Terms

Algorithm: A well-defined set of steps or processes used to solve a problem.

Architecture: The physical structure of a computer, its internal components (registers, memory, instruction set, input/output structure, etc.) and the way in which they interact.

Complex instruction set computer (CISC): A design style where there are few constraints on the types of instructions or addressing modes.

Context: The information needed to execute a process including the contents of program counters, address and data registers, and stack pointers

Cycles per instruction (CPI): A performance measurement used to judge the efficiency of a particular design.

Multithreaded: Several instruction streams or threads execute simultaneously.

Pipeline: Dividing up an instruction execution into overlapping steps that can be executed simultaneously with other instructions, each step represents a different pipeline stage.

Reduced instruction set computer (RISC): A design style where instructions are implemented for only the most frequently executed operations; addressing modes are limited to registers and special load/store modes to memory.

Superpipelined: A pipeline whose depth has been increased to allow more overlapped instruction execution.

Superscalar: Hardware capable of dynamically issuing more than one instruction per cycle.

Ternary logic: Digital logic with three valid voltage levels.

Very long instruction word (VLIW): Compiler techniques are used to concatenate many small instructions into a larger instruction word.

Wafer scale integration (WSI): Using an entire semiconductor wafer to implement a design without dicing the wafer into smaller chips.

© 2006 by Taylor & Francis Group, LLC

References

Anderson, T., Lazowska, E., and Levy, H. 1989. The performance implications of thread management alternatives for shared-memory multiprocessors. *IEEE Trans. on Computers* 38(12):1631–1644.

Flynn, M.J. 1966. Very high-speed computing systems. *Proc. of the IEEE* 54(12):1901–1909.

Hennessy, J. and Jouppi, N. 1991. Computer technology and architecture: An evolving interaction. *IEEE Computer* 24(9):18–29.

Hwang, K. 1993. *Advanced Computer Architecture: Parallelism, Scalability, Programmability*. McGraw-Hill, New York.

Keyes, R.W. 1991. The power of connections. *IEEE Circuits and Devices* 7(3):32–35.

Long, S.I. and Butner, S.E. 1990. *Gallium Arsenide Digital Integrated Circuit Design*. McGraw-Hill, New York.

Smith, B. 1978. A pipelined, shared resource MIMD computer. *International Conference on Parallel Processing*, (Bellaire, MI), pp. 6–8. (Aug).

Further Information

Books:

Baron R.J. and Higbee, L. 1992. *Computer Architecture*. Addison-Wesley, Reading, MA.

Mano, M.M. 1993. *Computer System Architecture*. Prentice-Hall, Englewood Cliffs, NJ.

Patterson, D.A. and Hennessy, J.L. 1991. *Computer Organization and Design: The Hardware/Software Interface*. Morgan-Kaufmann, San Mateo, CA.

Conference Proceedings:

International Conference on Parallel Processing
International Symposium on Computer Architecture
Proceedings of the International Conference on VLSI Design
Supercomputing

Magazines and Journals:

ACM Computer Architecture News
Computer Design
IEEE Computer
IEEE Micro
IEEE Transactions on Computers

© 2006 by Taylor & Francis Group, LLC

17

Software Design and Development

Margaret H. Hamilton

17.1 Introduction 17-1
17.2 The Notion of Software 17-2
17.3 The Nature of Software Engineering 17-5
17.4 A New Approach 17-9
17.5 Apollo On-Board Flight Software Effort: Lessons Learned 17-10
17.6 Development Before the Fact 17-12
17.7 Development Before the Fact Theory 17-15
17.8 Process 17-15
17.9 Select the Right Paradigm and then Automate 17-17

17.1 Introduction

A computer system can be neatly compared with a biological entity called a superorganism. Composed of software, hardware, peopleware and their interconnectivity (such as the internet), and requiring all to survive, the silicon superorganism is itself a part of a larger superorganism, for example, the business. It could be a medical system including patients, drugs, drug companies, doctors, hospitals and health care centers; a space mission including the spacecraft, the laws of the universe, mission control and the astronauts; a system for researching genes including funding organizations, funds, researchers, research subjects, and genes; or a financial system including investors, money, politics, financial institutions, stock markets, and the health of the world economy.

Whether the business be government, academic, or commercial, the computer system, like its biological counterpart, must grow and adapt to meet fast changing requirements. Like other organisms, the business has both physical infrastructure and operational policies which guide, and occasionally constrain, its direction and the rate of evolution, which it can tolerate without becoming dysfunctional.

Unlike a biological superorganism, which may take many generations to effect even a minor hereditary modification, software is immediately modifiable and is, therefore, far superior in this respect to the biological entity in terms of evolutionary adaptability. Continuity of business rules and the physical infrastructure provides a natural tension between "how fast the software can change" and "how rapidly the overall system can accept change."

As the brain of the silicon superorganism, software controls the action of the entire entity, keeping in mind, however, it was a human being that created the software.

In this chapter we will discuss the tenets of software, what it is and how it is developed, as well as the precepts of software engineering, which are the *methodologies* by which ideas are turned into software.

© 2006 by Taylor & Francis Group, LLC

17.2 The Notion of Software

Software is the embodiment of logical processes, whether in support of business functions or in control of physical devices. The nature of software as an instantiation of process can apply very broadly, when modeling complex organizations, or very narrowly as when implementing a discrete numerical algorithm. In the former case, there can be significant linkages between re-engineering businesses to accelerate the rate of evolution—even to the point of building operational models which then transition into application suites and thence into narrowly focused implementation of algorithms. Software thus has a potentially wide range of application, and when well designed, it has a potentially long period of utilization.

Whereas some would define software as solely the code generated from programming language statements in the compilation process, a broader and more precise definition includes requirements, specifications, designs, program listings, documentation, procedures, rules, measurements, and data, as well as the tools and reusables used to create, test, optimize, and implement the software.

That there is more than one definition of software is a direct result of the confusion about the very process of software development itself. A 1991 study by the Software Engineering Institute (SEI) (1991) amplifies this rather startling problem. The SEI developed a methodology for classifying an organization's software process maturity into one of five levels that range from Level 1, the *initial level* (where there is no formalization of the software process), to Level 5, the *optimizing level* where methods, procedures, and metrics are in place with a focus toward continuous improvement in software reliability. The result of this study showed that fully 86% of organizations surveyed in the US were at Level 1 where the terms "ad hoc," "dependent on heroes," and "chaotic" are commonly applied. And, given the complexity of today's applications including those that are internet based, it would not be surprising to see the percentage of Level 1 organizations increase. Adding to the mix are the organizations that think the so called order mandated by some techniques serves only to bring about more chaos, confusion, and complexity; or at least spend many more dollars for what they believe delivers little, no, or negative benefit.

Creating order from this chaos requires an insightful understanding into the component parts of software as well as the development process. Borrowing again from the world of natural science, an entelechy is something complex that emerges when you put a large number of simple objects together. For example, one molecule of water is rather boring in its utter lack of activity. But pour a bunch of these molecules into a glass and you have a ring of ripples on the water's surface. If you combine enough of these molecules together, you wind up with an ocean. So, too, software: by itself, a line of code is a rather simple thing. But combine enough of them together and you wind up with the complexity of a program. Add additional programs and you wind up with a system that can put a person on the moon.

Although the whole is indeed bigger than the sum of its parts, one must still understand those parts if the whole is to work in an orderly and controlled fashion. Like a physical entity, software can "wear" as a result of maintenance, changes in the underlying system and updates made to accommodate the requirements of the ongoing user community. Entropy is a significant phenomenon in software, especially for Level 1 organizations.

Software at the lowest programming level is termed *source code*. This differs from *executable code* (i.e., that which can be executed directly by the hardware to perform one or more specified functions) in that it is written in one or more programming languages and cannot, by itself, be executed by the hardware until it is translated into machine executable code. A programming language is a set of words, letters, numerals, and abbreviated mnemonics, regulated by a specific syntax, used to describe a program (made up of lines of source code) to a computer. There are a wide variety of programming languages, many of them tailored for a specific type of application. C, one of today's more popular programming languages, is used in engineering as well as business environments, whereas object-oriented languages, such as C++ (Stroustrup, 1997) and Java (Gosling et al., 1996), have been gaining acceptance in both of these environments. In fact Java has become a language of choice for internet based applications. In the recent past, engineering applications have often used programming languages such as Fortran, Ada (for government applications) and HAL (for space applications) (Lickly, 1974), while commercial business applications have favored common business oriented language (COBOL). For the most part one finds that in any given organization

© 2006 by Taylor & Francis Group, LLC

there are no prescribed rules that dictate which languages are to be used. As one might expect, a wide diversity of languages is being deployed.

The programming language, whether that be C (Harbison, 1997), Java, COBOL, C++, C# or something else, provides the capability to code such logical constructs as that having to do with:

- *User interface:* Provides a mechanism whereby the ultimate end user can input, view, manipulate, and query information contained in an organization's computer systems. Studies have shown that productivity increases dramatically when visual user interfaces are provided. Known as *graphical user interfaces (GUIs)*, each operating system provides its own variation. Some common graphical standards are Motif for Unix (including Linux) systems, Aqua for the Macintosh Operating System and Microsoft Windows for PC-based systems.
- *Model calculations:* Perform the calculations or algorithms (step-by-step procedures for solving a problem) intended by a program, for example, process control, payroll calculations, or a Kalman Filter.
- *Program control:* Exerts control in the form of comparisons, branching, calling other programs, and iteration to carry out the logic of the program.
- *Message processing:* There are several varieties of message processing. Help-message processing is the construct by which the program responds to requests for help from the end user. Error-message processing is the automatic capability of the program to notify and then recover from an error during input, output, calculations, reporting, communications, etc. And, in object-oriented development environments, message processing implies the ability of program objects to pass information to other program objects.
- *Moving data:* Programs store data in a data structure. Data can be moved between data structures within a program; moved from an external database or file to an internal data structure or from user input to a program's internal data structures. Alternatively, data can be moved from an internal data structure to a database or even to the user interface of an end user. Sorting, searching, and formatting are data moving and related operations used to prepare the data for further operations.
- *Database:* A collection of data (objects) or information about a subject or related subjects, or a system (for example, an engine in a truck or a personnel department in an organization). A database can include objects such as forms and reports or a set of facts about the system (for example, the information in the personnel department needed about the employees in the company). A database is organized in such a way as to be easily accessible to computer users. Its data is a representation of facts, concepts, or instructions in a manner suitable for processing by computers. It can be displayed, updated, queried, and printed, and reports can be produced from it. A database can store data in several ways including in a relational, hierarchical, network, or object-oriented format.
- *Data declaration:* Describes the organization of a data structure to the program. An example would be associating a particular data structure with its type (for example, data about a particular employee might be of type person).
- *Object:* A person, place, or thing. An object contains data and a set of operations to manipulate data. When brought to life, it knows things (called *attributes*) and can do things (to change itself or interact with other objects). For example, in a robotics system an object may contain the functions to move its own armature to the right, while it is coordinating with another robot to transfer yet another object. Objects can communicate with each other through a communications medium (e.g., message passing, radio waves, internet).
- *Real time:* A software system that satisfies critical timing requirements. The correctness of the software depends on the results of computation, as well as on the time at which the results are produced. Real-time systems can have varied requirements such as performing a task within a specific deadline and processing data in connection with another process outside of the computer. Applications such as transaction processing, avionics, interactive office management, automobile systems, and video games are examples of real-time systems.

© 2006 by Taylor & Francis Group, LLC

- *Distributed system:* Any system in which a number of independent interconnected processes can cooperate (for example processes on more than one computer). The client/server model is one of the most popular forms of distribution in use today. In this model, a client initiates a distributed activity and a server carries out that activity.
- *Simulation:* The representation of selected characteristics of the behavior of one physical or abstract system by another system. For example, a software program can simulate an airplane, an organization, a computer, or another software program.
- *Documentation:* Includes description of requirements, specification and design; it also includs comments that describe the operation of the program that is stored internally in the program, as well as written or generated documentation that describes how each program within the larger system operates.
- *Tools:* The computer programs used to design, develop, test, analyze, or maintain system designs of another computer program and its documentation. They include code generators, compilers, editors, *database management systems (DBMS)*, GUI builders, simulators, debuggers, operating systems and software development, and systems engineering tools (some derived or repackaged from earlier tools that were referred to in the nineties as *computer aided software engineering (CASE)* tools), that combine a set of tools, including some of those listed above.

Although the reader should by now understand the dynamics of a line of source code, where that line of source code fits into the superorganism of software is dependent on many variables. This includes the industry the reader hails from as well as the software development paradigm used by the organization.

As a base unit, a line of code can be joined with other lines of code to form many things. In a traditional software environment many lines of code form a program, sometimes referred to as an application program or just plain application. But lines of source code by themselves cannot be executed. First, the source code must be run through a *compiler* to create *object code*. Next, the object code is run through a *linker*, which is used to construct *executable code*. Compilers are programs themselves. Their function is twofold. The compiler first checks the source code for obvious syntax errors and then, if it finds none, creates the object code for a specific operating system. Unix, Linux (a spinoff of Unix), and Windows are all examples of operating systems. An operating system can be thought of as a supervising program that manages the application programs that run under its control. Since operating systems (as well as computer architectures) can be different from each other, compiled source code for one operating system cannot be executed under a different operating system, without a recompilation.

Solving a complex business or engineering problem often requires more than one program. One or more programs that run in tandem to solve a common problem is known collectively as an application system (or system). The more modern technique of object-oriented development dispenses with the notion of the program altogether and replaces it with the concept of an object (Goldberg and Robson, 1983; Meyer and Bobrow, 1992; Stefik and Bobrow, 1985; Stroustrup, 1994).

Where a program can be considered a critical mass of code that performs many functions in the attempt to solve a problem with little consideration for object boundaries, an object is associated with the code to solve a particular set of functions having to do with just that type of object. By combining objects like molecules, it is possible to create more efficient systems than those created by traditional means. Software development becomes a speedier and less error-prone process as well. Since objects can be reused, once tested and implemented, they can be placed in a library for other developers to reuse. The more the objects in the library, the easier and quicker it is to develop new systems. And since the objects being reused have, in theory, already been warranted (i.e., they have been tested and made error free), there is less possibility that object-oriented systems will have major defects.

The process of building programs and/or objects is known as software development, or software engineering. It is composed of a series of steps or phases, collectively referred to as a development life cycle. The phases include (at a bare minimum): an analysis or requirements phase where the business or engineering problem is dissected and understood, a specification phase where decisions are made as to how the requirements will be fulfilled (e.g., deciding what functions are allocated to software and what functions are

© 2006 by Taylor & Francis Group, LLC

allocated to hardware); a design phase where everything from the GUI to the database to the algorithms to the output is designed; a programming (or implementation) phase where one or more tools are used to support the manual process of coding or to automatically generate code, a testing (debugging) phase where the code is tested against a business test case and errors in the program are found and corrected, an installation phase where the systems are placed in production, and a maintenance phase where modifications are made to the system. But different people develop systems in different ways. These different paradigms make up the opposing viewpoints of software engineering.

17.3 The Nature of Software Engineering

Engineers often use the term systems engineering to refer to the tasks of specifying, designing, and simulating a non-software system such as a bridge or electronic component. Although software may be used for simulation purposes, it is but one part of the systems engineering process. Software engineering, on the other hand, is concerned with the production of nothing but software. In the 1970s industry pundits began to notice that the cost of producing large-scale systems was growing at a high rate and that many projects were failing or, at the very least, resulting in unreliable products. Dubbed the software crisis, its manifestations were legion, and the most important include

- *Programmer productivity:* In government in the 1980s, an average developer using C was expected to produce only 10 lines of production code per day (an average developer within a commercial organization was expected to produce 30 lines a month); today the benchmark is more like two to five lines a day while at the same time the need is dramatically higher than that, perhaps by several orders of magnitude, ending up with a huge backlog. The National Software Quality Experiment (NSQE) completed a 10-year research project in 2002 that compared the productivity of systems today with that of systems 10 years ago. The assumption at the outset was that the productiviity of today's systems would be much higher given all the more "modern" tools including a proliferation of commercial packages and object-oriented languages that are now available; but they were wrong. Not only did the productivity not increase, it decreased (NSQE, 2002). Further NSQE concluded that there would need to be a significant breakout in the way software is designed and developed in order to achieve a factor of 10 reduction in defect rate.

  Programmer productivity is dependent on a plethora of vagaries: from expertise to complexity of the problem to be coded to size of the program that is generated (Boehm, 1981). The science of measuring the quality and productivity of the software engineering process is called *metrics.* As in the diverse paradigms in software engineering itself, there are many paradigms of software measurement. Today's metric formulas are complex and often take into consideration the following: cost, time to market, productivity on prior projects, data communications, distributed functions, performance, heavily used configurations, transaction rates, online data entry, end user efficiency, online update, processing complexity, reusability, installation ease, operational ease, and multiplicity of operational sites.
- *Defect removal costs:* The same variables that affect programmer productivity affect the cost of debugging the programs and objects generated by those programmers. It has been observed that the testing and correcting of programs consumes a large share of the overall effort.
- *Development environment:* Development tools and development practices greatly affect the quantity and quality of software. Most of today's design and programming environments contain only a fragment of what is really needed to develop a complete system. Life cycle development environments provide a good example of this phenomena. Most of these tools can be described either as addressing the upper part of the life cycle (i.e., they handle the analysis and design) or the lower part of the life cycle (i.e., they handle code generation). There are few integrated tools on the market (i.e., that seamlessly handle both upper and lower functionalities). There are even fewer tools that add simulation, testing and cross-platform generation to the mix. And it would be hard put to find any tools that seamlessly integrate system design to software development.

© 2006 by Taylor & Francis Group, LLC

- *GUI development:* Developing graphical user interfaces is a difficult and expensive process unless the proper tools are used. The movement of systems from a host-based environment to the workstation PC saw the entry of a plethora of GUI development programs onto the marketplace. But, unfortunately, the vast majority of these GUI-based tools do not have the capability of developing the entire system (i.e., the processing component as opposed to merely the front end). This leads to fragmented and error-prone systems. To be efficient, the GUI builder must be well integrated into the software development environment.

The result of these problems is that most of today's systems require more resources allocated to maintenance than to their original development. Lientz and Swanson (1980) demonstrate that the problem is, in fact, larger than the one originally discerned during the 1970s. Software development is indeed complex and the limitations on what can be produced by teams of software engineers given finite amounts of time, budgeted dollars, and talent have been amply documented (Jones, 1977).

Essentially the many paradigms of software engineering attempt to rectify the causes of declining productivity and quality. Unfortunately this fails because current paradigms treat symptoms rather than the root problem. In fact, software engineering is itself extremely dependent on both the underlying systems software (for example, the operating systems) and hardware as well as the business environments upon which they sit.

SEI's process maturity grid very accurately pinpoints the root of most of our software development problems. The fact that a full 86% of the organizations studied, remain at the ad hoc or chaotic level indicates that only a few organizations (the remaining 14%) have adopted any formal process for software engineering. Simply put, 86% of all organizations react to a business problem by just writing code. If they do employ a software engineering discipline, in all likelihood, it is one that no longer fits the requirements of the ever evolving business environment.

In the 1970s, the structured methodology was popularized. Although there were variations on the theme (i.e., different versions of the structured technique included the popular Gane/Sarson method and Yourdon method), for the most part, it provided methodology to develop usable systems in an era of batch computing. In those days, on-line systems with even the dumbest of terminals were a radical concept and graphical user interfaces were as unthinkable as the fall of the Berlin Wall.

Although times have changed and today's hardware is a thousand times more powerful than when structured techniques were introduced, this technique still survives. And survives in spite of the fact that the authors of these techniques have moved on to more adaptable paradigms, and more modern software development and systems engineering environments have entered the market.

In 1981 Finkelstein and Martin (Martin, 1981) popularized information engineering for the more commercially oriented users (i.e., those whose problems to be solved tended to be more database centered), which, to this day, is quite popular amongst mainframe developers with an investment in CASE. Information engineering is essentially a refinement of the structured approach. Instead of focusing on the data so preeminent in the structured approach, however, information engineering focuses on the information needs of the entire organization. Here business experts define high-level information models, as well as detailed data models. Ultimately the system is designed from these models.

Both structured and information engineering methodologies have their roots in mainframe-oriented commercial applications. Today's migration to client/server technologies (where the organization's data can be spread across one or more geographically distributed servers while the end-user uses his or her GUI of choice to perform local processing) disables most of the utility of these methodologies. In fact, many issues now surfacing in more commercial applications are not unlike those addressed earlier in the more engineering oriented environments such as tele-communications and avionics. Client/server environments are characterized by their diversity. One organization may store its data on multiple databases, program in several programming languages, and use more than one operating system and, hence, different GUIs. Since software development complexity is increased a hundredfold in this new environment, a better methodology is required. Today's object-oriented techniques solve some of the problem. Given the complexity of client/server, code trapped in programs is not flexible enough to meet the needs of this type of environment. We have already discussed how coding via objects rather than large programs engenders

© 2006 by Taylor & Francis Group, LLC

flexibility as well as productivity and quality through reusability. But object-oriented development is a double-edged sword.

While it is true that to master this technique is to provide dramatic increases in productivity, the sad fact of the matter is that object-oriented development, if done inappropriately, can cause problems far greater than problems generated from structured techniques. The reason for this is simple: the stakes are higher. Object-oriented environments are more complex than any other, the business problems chosen to be solved by object-oriented techniques are far more complex than other types of problems, and there are no conventional object-oriented methodologies and corollary tools to help the development team develop systems that are truly reliable. There are many kinds of object orientation. But with this diversity comes some very real risks. As a result, the following developmental issues must be considered before the computer is even turned on.

Integration is a challenge and needs to be considered at the onset. With traditional systems, developers rely on mismatched modeling methods to capture aspects of even a single definition. Whether it be integration of object to object, module to module, phase to phase, or type of application to type of application, the process can be an arduous one. The mismatch of evolving versions of point products (where a point product is able to be used for developing only an aspect of an application) used in design and development compounds the issue. Integration is usually left to the devices of myriad developers well into development. The resulting system is sometimes hard to understand and objects are difficult to trace. The biggest danger is that there is little correspondence to the real world. Interfaces are often incompatible and errors propagate throughout development. As a result, systems defined in this manner can be ambiguous and just plain incorrect.

Errors need to be minimized. Traditional methods as well as traditional object-oriented methods actually encourage the propagation of errors, such as reuse of unreliable objects with embedded and inherited errors. Internet viruses, worms, packet storms, trojans, and other malicious phenomenon are inexcusable and should be the exception not the rule. To be successful, errors must be eliminated from the very onset of the development process, including in the operating systems and related systems.

Languages need to be more formal. Whereas some languages are formal and others friendly, it is hard to find a language both *formal* and *friendly*. Within environments where more informal languages are used, lack of traceability and an overabundance of interface errors are a common occurrence. Within environments that have more formal languages their application has been found to be impractical for real world systems of any size or complexity. Recently, more modern software requirements languages have been introduced (for example, the Unified Modeling Language, UML (Booch et al., 1999), most of which are informal (or semi-formal); some of these languages were created by "integrating" several languages into one. Unfortunately, the bad comes with the good—often, more of what is not needed and less of what is needed; and since much of the formal part is missing, a common semantics needs to exist to reconcile differences and eliminate redundancies.

Flexibility for change and handling the unpredictable needs to be dealt with up front. Too often it is forgotten that the building of an application must take into account its evolution. In the real world, users change their minds, software development environments change and technologies change. Definitions of requirements in traditional development scenarios concentrate on the application needs of the user, but without consideration of the potential for the user's needs or environment to change. Porting to a new environment becomes a new development for each new architecture, operating system, database, graphics environment, language, or language configuration. Because of this, critical functionality is often avoided for fear of the unknown, and maintenance, the most risky and expensive part of a system's life cycle, is left unaccounted for during development. To address these issues, tools and techniques must be used to allow cross technology and changing technology solutions as well as provide for changing and evolving architectures.

The syndrome of locked-in design needs to be eliminated. Often, developers are forced to develop in terms of an implementation technology that does not have an open architecture, such as a specific database schema or a graphical user interface (GUI). Bad enough is to attempt an evolution of such a system; worse yet is to use parts of it as reusables for a system that does not rely on those technologies. Well thought out and formal business practices and their implementation will help to minimize this problem within an organization.

© 2006 by Taylor & Francis Group, LLC

Developers must prepare for parallelism and distributed environments. Often, when it is not known that a system is targeted for a distributed environment, it is first defined and developed for a single processor environment and then changed/redeveloped for a distributed environment—an unproductive use of resources. Parallelism and distribution need to be dealt with at the very start of the project.

Resource allocation should be transparent to the user. Whether or not a system is allocated to distributed, asynchronous, or synchronous processors and whether or not two processors or ten processors are selected; with traditional methods it is still up to the designer and developer to be concerned with manually incorporating such detail into the application. There is no separation between the specification of what the system is to do vs. how the system does it. This results in far too much implementation detail to be included at the level of design. Once such a resource architecture becomes obsolete, it is necessary to redesign and redevelop those applications which have old designs embedded within them.

The creation of reliable reusable definitions must be promoted, especially those that are inherently provided. Traditional requirements definitions lack the facilities to help find, create, use and ensure commonality in systems. Modelers are forced to use informal and manual methods to find ways to divide a system into components natural for reuse. These components do not lend themselves to integration and, as a result, they tend to be error-prone. Because these systems are not portable or adaptable, there is little incentive for reuse. In conventional methodologies, redundancy becomes a way of doing business. Even when methods are object oriented, developers are often left to their own devices to explicitly make their applications become object oriented; because these methods do not support all that which is inherent in an object.

Automation that minimizes manual work needs to replace "make work" automated solutions. In fact, automation itself is an inherently reusable process. If a system does not exist for reuse, it certainly does not exist for automation. But most of today's development process is needlessly manual. Today's systems are defined with insufficient intelligence for automated tools to use them as input. In fact, these automated tools concentrate on supporting the manual process instead of doing the real work. Typically, developers receive definitions that they manually turn into code. A process that could have been mechanized once for reuse is performed manually again and again. Under this scenario, even when automation attempts to do the real work, it is often incomplete across application domains or even within a domain, resulting in incomplete code, such as shell code (code that is an outline for real code). The generated code is often inefficient or hard wired to an architecture, a language, or even a version of a language. Often partial automations need to be integrated with incompatible partial automations or manual processes. Manual processes are needed to complete unfinished automations.

Run-time performance analysis (decisions between algorithms or architectures) should be based on formal definitions. Conventional system definitions contain insufficient information about a system's run-time performance, including that concerning the decisions between algorithms or architectures. Design decisions where this separation is not taken into account thus depend on analysis of outputs from ad hoc implementations and associated testing scenarios. System definitions must consider how to separate the target system from its target environment.

Design integrity is the first step to usable systems. It is not known if a design is a good one until its implementation has failed or succeeded. Spiral development (an evolutionary approach) as opposed to waterfall development (where each phase of development is completed before the next phase is begun) helps address some of this issue. Usually, a system design is based on short-term considerations because knowledge is not reused from previous lessons learned. Development, ultimately, is driven toward failure.

Once issues like these are addressed, software costs less and takes less time to develop. But, time is of the essence. These issues are becoming more complex and even more critical as developers prepare for the distributed environments that go hand in hand with the increasing predominance of the internet.

Thus far this chapter has explained the derivation of software and attempted to show how it has evolved over time to become the true brains of any automated system. But like a human brain, this software brain must be carefully architected to promote productivity, foster quality, and enforce control and reusability.

Traditional software engineering paradigms fail to see the software development process from the larger perspective of the superorganism described at the beginning of this chapter. It is only when we see the

© 2006 by Taylor & Francis Group, LLC

software development process as made up of discrete but well-integrated components, can we begin to develop a methodology that can produce the very benefits that have been promised by the advent of software decades ago.

Software engineering, from this perspective, consists of a methodology as well as a series of tools with which to implement the solution to the business problem at hand. But even before the first tool can be applied, the software engineering methodology must be deployed to assist in specifying the requirements of the problem. How can this be accomplished successfully in the face of the problems outlined? How can it be accomplished in situations where organizations must develop systems that run across diverse and distributed hardware platforms, databases, programming languages, and GUIs when traditional methodologies make no provision for such diversity? And how can software be developed without having to fix or cure those myriad of problems that result "after the fact" of that software's development?

To address these software issues, an organization has several options, ranging from one extreme to the other. The options include (1) keep doing things the same way; (2) add tools and techniques that support business as usual but provide relief in selected areas; (3) bring in more modern but traditional tools and techniques to replace existing ones; (4) use a new paradigm with the most advanced tools and techniques that formalizes the process of software development, while at the same time capitalizing on software already developed; or (5) completely start over with a new paradigm that formalizes the process of software development and uses the most advanced tools and techniques.

17.4 A New Approach

The typical reaction to the well known problems and challenges of software and its development has been to lament the fact that this is the way software is and to accept it as a "*fait acccompli*." But, such a reaction is unacceptable. What is required is a radical revision of the way we build software—an approach that understands how to build systems using the right techniques at the right time. First and foremost, it is a preventative approach. This means it provides a framework for doing things in the right way the first time. Problems associated with traditional methods of design and development would be prevented "before the fact" just by the way a system is defined. Such an approach would concentrate on preventing problems of development from even happening; rather than letting them happen "after the fact" and fixing them after they have surfaced at the most inopportune and expensive point in time.

Consider such an approach in its application to a human system. To fill a tooth before it reaches the stage of a root canal is curative with respect to the cavity, but preventive with respect to the root canal. Preventing the cavity by proper diet prevents not only the root canal, but the cavity as well. To follow a cavity with a root canal is the most expensive alternative; to fill a cavity on time is the next most expensive; and to prevent these cavities in the first place is the least expensive option. Preventiveness is a relative concept. For any given system, be it human or software, one goal is to prevent, to the greatest extent and as early as possible, anything that could go wrong in the life cycle process.

With a preventative philosophy, systems would be carefully constructed to minimize development problems from the very outset. A system that could be developed with properties that controlled its very own design and development. One result would be reusable systems that promote automation. Each system definition would model both its application and its life cycle with built-in constraints—constraints that protect the developer, but yet do not take away his flexibility.

Such an approach could be used throughout a life cycle, starting with requirements and continuing with functional analysis, simulation, specification, analysis, design, system architecture design, algorithm development, implementation, configuration management, testing, maintenance, and reverse engineering. Its users would include end users, managers, system engineers, software engineers, and test engineers.

With this approach, the same language would be used to define any aspect of any system and integrate it with any other aspect. The crucial point is that these aspects are directly related to the real world and, therefore, the same language would be used to define system requirements, specifications, design, and detailed design for functional, resource, and resource allocation architectures throughout all levels and layers of seamless definition, including hardware, software, and peopleware. The same language would be

© 2006 by Taylor & Francis Group, LLC

used to define organizations of people, missile or banking systems, cognitive systems, as well as real-time or database environments and is therefore appropriate across industries, academia, or government.

The philosophy behind preventative systems is that reliable systems are defined in terms of reliable systems. Only reliable systems are used as building blocks, and only reliable systems are used as mechanisms to integrate these building blocks to form a new system. The new system becomes a reusable for building other systems.

Effective reuse is a preventative concept. That is, reusing something (e.g., requirements or code) that contains no errors to obtain a desired functionality avoids both the errors and the cost of developing a new system. It allows one to solve a given problem as early as possible, not at the last moment. But to make a system truly reusable, one must start not from the customary end of a life cycle, during the implementation or maintenance phase, but from the very beginning.

Preventative systems are the true realization of the entelechy construct where molecules of software naturally combine to form a whole, which is much greater than the sum of its parts. Or one can think of constructing systems from the tinker toys of our youth. One recalls that the child never errs in building magnificent structures from these tinkertoys. Indeed, tinker toys are built from blocks that are architected to be perpetually reusable, perfectly integratable, and infinitely user-friendly.

There is at least one approach that follows a preventative philosophy. Although not in the mainstream yet, it has been used successfully by research and "trail blazer" organizations and is now being adopted for more commercial use. With this approach there is the potential for users to design systems and build software with seamless integration, including from systems to software, with: no longer a need for the system designer or system engineer to understand the details of programming languages or operating systems; no interface errors; defect rates reduced by at least a factor of 10; correctness by built-in language properties; unambiguous requirements, specifications, design (removing complexity, chaos and confusion); guarantee of function integrity after implementation; complete traceability and evolvability (application to application, architecture to architecture, technology to technology); maximized reuse; generation of complete software solutions from system specifications (full life cycle automation including 100% production ready code for any kind or size of system); a set of tools, each of which is defined and automatically generated by itself; significantly higher reliability, higher productivity, and lower risk.

Most people would say this is not possible, at least in the foreseeable future. This is because it has not been possible with traditional environments. It is possible, however, in major part, because of a non-traditional systems design and software development paradigm and its associated universal systems language that has been derived and evolved over three decades from an empirical study of the Apollo on-board flight software effort. In addition to experience with Apollo and other real world systems, this paradigm also takes its roots from systems theory, formal methods, formal linguistics, and object technologies. We'll describe this technology and its background in order to illustrate by example the potential that preventative approaches have.

17.5 Apollo On-Board Flight Software Effort: Lessons Learned

The original purpose of the empirical study, which had its beginnings in 1968, was to learn from Apollo' s flight software and its development in order to make use of this effort for future Apollo missions as well as for the then up and coming Skylab and Shuttle missions. A better way was needed to define and develop software systems than the ones being used and available, because the existing ones (just like the traditional ones today) did not solve the pressing problems. There was a driving desire to learn from experience; what could be done better for future systems and what should the designers and developers keep doing because they were doing it right. The search was in particular for a means to build ultra reliable software.

The results of the study took on multiple dimensions, not just for space missions but for systems in general; some of which were not so readily apparent for many years to come. The Apollo software, given what had to be accomplished and the time within which it had to be accomplished, was as complex as it could get; causing other software projects in the future to look (and be) less daunting than they might have been in comparison. The thought was if problems from a worst case scenario could be solved and simplified, the solutions might be able to be applied to all kinds of systems.

© 2006 by Taylor & Francis Group, LLC

On hindsight, this was an ideal setting from which to understand systems and software and the kinds of problems that can occur and issues that need to be addressed. Because of the nature of the Apollo software there was the opportunity to make just about every kind of error possible, especially since the software was being developed concurrently with the planning of the space missions, the hardware, the simulator, and the training of the astronauts (demonstrating how much the flight software was part of a larger system of other software, hardware and peopleware); and since no one had been to the moon before there were many unknowns. In addition the developers were under the gun with what today would have been unrealistic expectations and schedules. This and what was accomplished (or not accomplished) provided a wealth of information from which to learn. Here are some examples:

It was found during the Apollo study that interface errors (errors resulting from ambiguous relationships, mismatches, conflicts in the system, poor communication, lack of coordination, inability to integrate) accounted for approximately 75% of all the errors found in the software during final testing, the Verification and Validation (V&V) phase (using traditional development methods the figure can be as high as 90%); interface errors include data flow, priority and timing errors from the highest levels of a system to the lowest levels of a system, to the lowest level of detail. It was also determined that 44% of the errors were found by manual means (suggesting more areas for automation) and that 60% of the errors had unwittingly existed in earlier missions—missions that had already flown, though no errors occurred (made themselves known) during any flights. The fact that this many errors existed in earlier missions was down-right frightening. It meant lives were at stake during every mission that was flown. It meant more needed to be done in the area of reliability. Although no software problem ever occurred (or was known to occur) on any Apollo mission, it was only because of the dedication of the software team and the methods they used to develop and test the software.

A more detailed analysis of interface errors followed; especially since they not only accounted for the majority of errors but they were often the most subtle errors and therefore the hardest to find. The realization was made that integration problems could be solved if interface problems could be solved and if integration problems could be solved, there would be traceability. Each interface error was placed into a category according to the means that could have been taken to prevent it by the very way a system was defined. It was then established that there were ways to prevent certain errors from happening simply by changing the rules of definition. This work led to a theory and methodology for defining a system that would eliminate all interface errors.

It quickly became apparent that everything including software is a system and the issues of system design were one and the same as those of software. Many things contributed to this understanding of what is meant by the concept of "system." During development of the flight software, once the software group implemented the requirements ("thrown over the wall" by system design experts to the software people), the software people necessarily became the new experts. This phenomenon forced the software experts to become system experts (and vice versa) suggesting that a system was a system whether in the form of higher level algorithms or software that implemented those algorithms.

Everyone learned to always expect the unexpected and this was reflected in every design; that is, they learned to think, plan and design in terms of error detection and recovery, reconfiguration in real time; and to always have backup systems. The recognition of the importance of determining and assigning (and ensuring) unique priorities of processes was established as part of this philosophy. Towards this end it was also established that a "kill and start over again" restart approach to error detection and recovery was far superior as opposed to a "pick up from where you left off" approach; and it simplified both the operation of the software as well as the development and testing of the software. The major impact a system design for one part of the software could have on another part further emphasized that everything involved was part of a system—the "what goes around comes around" syndrome. For example, choosing an asynchronous executive (one where higher priority processes can interrupt lower priority processes) for the flight software not only allowed for more flexibility during the actual flights but it also provided for more flexibility to make a change more safely and in a more modular fashion during development of the flight software.

© 2006 by Taylor & Francis Group, LLC

It became obvious that testing was never over (even with many more levels of testing than with other kinds of applications; and with added testing by independent verification and validation organizations). This prompted a search for better methods of testing. Again, traditional methods were not solving the problem.

With the realization that most of the system design and software development processes could be mechanized (and the same generic processes were being performed throughout each of all phases of development), it became clear they could be automated. This suggested an environment that would do just that, automate what traditional systems still to this day do manually. In addition, for the space missions, software was being developed for many missions all at once and in various stages, leading the way to learn how to successfully perform distributed development.

It was learned during this process of evaluation of space flight software that traditional methods did not support developers in many areas and allowed too much freedom in other areas (such as freedom to make errors, both subtle and serious) and not enough freedom in other areas (such as areas that required an open architecture and the ability to reconfigure in real time to be successful). It further became clear that a new kind of language was needed to define a system having certain "built-in" properties not available in traditional languages; such as inherent reliability, integration, reuse, and open architecture capabilities provided simply by the use of the language itself. In addition it was realized a set of tools could be developed to support such a language as well as take away tedious mechanical processes that could become automated, thus avoiding even further errors, reducing the need for much after the fact testing. It was only later understood the degree to which systems with "built-in reliability" could increase the productivity in their development, resulting in "built-in productivity."

Lessons learned from this effort continue today. Key aspects are that systems are asynchronous in nature and this should be reflected inherently in the language used to define systems. Systems should be assumed to be event driven and every function to be applied should have a unique priority; real time event and priority driven behavior should be part of the way one specifies a system in a systems language and not defined on a case by case basis and in different programming languages with special purpose data types.

Rather than having a language whose purpose is to run a machine the language should naturally describe what objects there are and what actions they are taking. Objects are inherently distributed and their interactions asynchronous with real time, event-driven behavior. This implies that one could define a system and its own definition would have the necessary behaviors to characterize natural behavior in terms of real time execution semantics. Application developers would no longer need to explicitly define schedules of when events were to occur. Events would instead occur when objects interact with other objects. By describing the interactions between objects the schedule of events is inherently defined.

The result of this revelation was that a universal systems language could be used to tackle all aspects of a system: its planning, development, deployment, and evolution. This means that all the systems that work together during the system's lifecycle can be defined and understood using the same semantics.

17.6 Development Before the Fact

Once the analysis of the Apollo effort was completed, the next step was to create (and evolve) a new mathematical paradigm from the "heart and soul" of Apollo; one that was preventative instead of curative in its approach. A theory was derived for defining a system such that the entire class of errors, known as interface errors, would be eliminated. The first generation technology derived from this theory concentrated on defining and building reliable systems in terms of functional hierarchies (Hamilton, 1986). Having realized the benefits of addressing one major issue, that is, reliability, just by the way a system is defined, the research effort

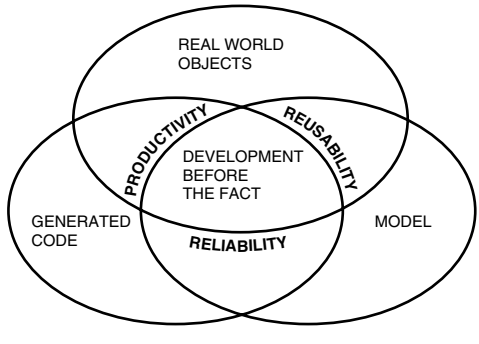


FIGURE 17.1 The development-before-the-fact paradigm.

© 2006 by Taylor & Francis Group, LLC

continued over many years (and still continues) to evolve the philosophy of addressing this issue further as well as addressing other issues the same way, that is, using language mechanisms that inherently eliminate software problems. The result is a new generation technology called development before the fact (DBTF) (see Fig. 17.1) where systems are designed and built with preventative properties integrating all aspects of a system's definition including the inherent integration of functional and type hierarchical networks, (Hamilton and Hackler, in press; Hamilton, 1994; Hamilton, 1994; Keyes, 2001), <http://www.htius.com/>.

Development before the fact is a system oriented object (SOO) approach based on a concept of control that is lacking in other software engineering paradigms. At the base of the theory that embodies every system are a set of axioms—universally recognized truths—and the design for every DBTF system is based on these axioms and on the assumption of a universal set of objects. Each axiom defines a relation of immediate domination. The union of the relations defined by the axioms is control. Among other things, the axioms establish the control relationships of an object for invocation, input and output, input and output access rights, error detection and recovery, and ordering during its developmental and operational states. Table 17.1 summarizes some of the properties of objects within these systems.

Combined with further research it became clear that the root problem with traditional approaches is that they support users in "fixing wrong things up rather than in "doing things the right way in the first place." Instead of testing software to look for errors *after* the software is developed, with the new paradigm, software could now be defined to not allow errors in, in the first place; correctness could be accomplished by the very way software is defined, by "built-in" language properties; what had been created was a universal semantics for defining not just software systems but systems in general.

TABLE 17.1 System Oriented Object Properties of Development Before the Fact

Quality
Reliable
Affordable

Reliable
In control and under control
Based on a set of axioms
  domain identification (intended, unintended)
  ordering (priority and timing)
  access rights: incoming object (or relation), outgoing object (or relation)
  replacement
Formal
  consistent, logically complete
  necessary and sufficient
  common semantic base
  unique state identification
Error free (based on formal definition of "error")
  always gets the right answer at the right time and in the right place
  satisfies users and developers intent
Handles the unpredictable
Predictable

Affordable
Reusable
Optimizes resources in operation and development
  in minimum time and space
  with best fit of objects to resources

Reusable
Understandable, integratable and maintainable
Flexible
Follows standards

Automation
Common definitions
  natural modularity
    natural separation (e.g., functional architecture from its resource architectures);
  dumb modules
  an object is integrated with respect to structure, behavior and properties of control
  integration in terms of structure and behavior
  type of mechanisms
    function maps (relate an object's function to other functions)
    object type maps (relate objects to objects)
    structures of functions and types
  category
    relativity
      instantiation
      polymorphism
      parent/child
      being/doing
      having/not having
    abstraction
      encapsulation
      replacement
      relation including function
      typing including classification
      form including both structure and behavior (for object types and functions)
    derivation
      deduction
      inference
      inheritance

(*continued*)

© 2006 by Taylor & Francis Group, LLC

TABLE 17.1 System Oriented Object Properties of Development Before the Fact (*Continued*)

Handles the unpredictable
Throughout development and operation
Without affecting unintended areas
Error detect and recover from the unexpected
Interface with, change and reconfigure in asynchronous, distributed, real time environment

Flexible
Changeable without side effects
Evolvable
Durable
Reliable
Extensible
Ability to break up and put together
- one object to many: modularity, decomposition, instantiation
- many objects to one: composition, applicative Operators, integration, abstraction

Portable
- secure
- diverse and changing layered developments
- open architecture (implementation, resource allocation, and execution independance)
- plug-in (or be plugged into) or reconfiguration of different modules
- adaptable for different organizations, applications, functionality, people, products

Automation
The ultimate form of reusable
Formalize, mechanize, then automate
- it
- its development
- that which automates its development

Understandable, integratable, and maintainable
Reliable
A measurable history
Natural correspondence to real world
- persistence, create and delete
- appear and disappear
- accessibility
- reference
- assumes existence of objects
- real time and space constraints
- representation
- *relativity, abstraction, derivation*

Provides user friendly definitions
- recognizes that one user's friendliness is another user's nightmare
- hides unnecessary detail (*abstraction*)
- variable, user selected syntax
- self teaching
- derived from a common semantic base
- common definition mechanisms

Communicates with common semantics to all entities
Defined to be simple as possible but not simpler
Defined with integration of all of its objects (and all aspects of these objects)
Traceability of behavior and structure and their changes (maintenance) throughout its birth, life, and death
Knows and able to reach the state of completion
- definition
- development of itself and that which develops it
  - analysis
  - design
  - implementation
  - instantiation
  - testing
  - maintenance

*All italicized words point to a reusable.

Once understood, it became clear that the characteristics of good design can be reused by incorporating them into a language for defining any system (not just a software system). The language is a formalism for representing the mathematics of systems. This language—actually a *meta*-language—is the key to DBTF. Its main attribute is to help the designer reduce the complexity and bring clarity into his thinking process, eventually turning it into the ultimate reusable, wisdom, itself. It is a universal systems language for defining systems, each system of which can be incorporated into the meta-language and then used to define other systems. A system defined with this language has properties that come along "for the ride" that in essence control its own destiny. Based on a theory (DBTF) that extends traditional mathematics of systems with a unique concept of control, this formal but friendly language has embodied within it a natural representation of the physics of time and space. 001AXES evolved as DBTF's formal universal systems language, and the 001 Tool Suite as its automation.

The creation of the concept of reuse definition scenarios during Apollo to save time in development and space in the software was a predecessor to the type of higher level language statements used within the systems language. This included horizontal and vertical reuse that led to a flexible open architecture within the DBTF environment.

Understanding the importance of metrics and the influence it could have on future software has had a major focus throughout this endeavor. What this approach represents, in essence, is the current state of a "reusable" originally based on and learned from Apollo followed by that learned from each of its

© 2006 by Taylor & Francis Group, LLC

evolving states. It reuses and builds on that which was (and is) observed to have been beneficial to inherit as well as the avoidance of that observed not to have been beneficial. That learned from history was and will continue to be put into practice for future projects. Someone once said "it is never surprising when something developed empirically turns out to have intimate connections with theory." Such is the case with DBTF .

17.7 Development Before the Fact Theory

Mathematical approaches have been known to be difficult to understand and use. They have also been known to be limited in their use for nontrivial systems as well as for much of the life cycle of a given system. What makes DBTF different in this respect is that the mathematics it is based on, has been extended for handling the class of systems that software falls into; the formalism along with its unfriendliness is "hidden" under the covers by language mechanisms derived in terms of that formalism; and the technology based on this formalism has been put to practical use.

With DBTF, all models are defined using its language (and developed using its automated environment) as SOOs. A SOO is understood the same way without ambiguity by all other objects within its environment—including all users, models, and automated tools. Each SOO is made up of other SOOs. Every aspect of a SOO is integrated not the least of which is the integration of its object oriented parts with its function oriented parts and its timing oriented parts. Instead of systems being object-oriented, objects are systems-oriented. All systems are objects. All objects are systems. Because of this, many things heretofore not believed possible with traditional methods are possible, much of what seems counter intuitive with traditional approaches, that tend to be software centric, becomes intiutive with DBTF, which is system centric.

DBTF's automation provides the means to support a system designer or software developer in following its paradigm throughout a system's design or software development life cycle. Take for example, testing. The more a paradigm prevents errors from being made in the first place the less the need for testing and the higher the productivity. Before the fact testing is inherently part of every development step. Most errors are prevented because of that which is inherent or automated.

Unlike a language such as UML or Java that is limited to software, 001AXES is a systems language and as such can be used to state not just software phenomenon but any given system phenomenon. UML is a graphical software specification language. 001AXES is a graphical system specification language. They have different purposes. At some point UML users have to program in some programming language, but 001AXES users do not; a system can be completely specified and automatically generated with all of its functionality, object and timing behavior without a line of code being written. The intent of UML is to run a machine. The intent of 001AXES is to define (and if applicable, execute) a system software or otherwise.

Because 001AXES is a systems language, its semantics is more general than a software language. For example, 001AXES defined systems could run on a computer, person, autonomous robot, or an organization; whereas the content of the types (or classes) in a software language is based on a computer that runs software.

Because 001AXES is systemic in nature it capitalizes upon that which is common to all systems; including software, financial, political, biological, and physical systems. For software the semantics of 001AXES is mapped to appropriate constructs for each of a possible set of underlying programming language implementations. Unlike the DBTF approach, in general these more traditional development approaches emphasize a fragmented approach to the integrated specification of a system and its software, short changing integration and automation in the development process. A more detailed discussion of 001AXES and UML can be found in (Hamilton and Hackler, 2000).

17.8 Process

Derived from the combination of steps taken to solve the problems of traditional systems engineering and software development, each DBTF system is defined with built-in quality, built-in productivity and built-in control (like the biological superorganism). The process combines mathematical perfection with

© 2006 by Taylor & Francis Group, LLC

engineering precision. Its purpose is to facilitate the "doing things right in the first place" development style, avoiding the "fixing wrong things up" traditional approach. Its automation is developed with the following considerations: error prevention from the early stage of system definition, life cycle control of the system under development, and inherent reuse of highly reliable systems. The development life cycle is divided into a sequence of stages, including: requirements and design modeling by formal specification and analysis; automatic code generation based on consistent and logically complete models; test and execution; and simulation.

The first step is to define a model with the language. This process could be in any phase of the developmental life cycle, including problem analysis, operational scenarios, and design. The model is automatically analyzed to ensure that it was defined properly. This includes static analysis for preventive properties and dynamic analysis for user intent properties. In the next stage, the generic source code generator automatically generates a fully production-ready and fully integrated software implementation for any kind or size of application, consistent with the model, for a selected target environment in the language and architecture of choice. If the selected environment has already been configured, the generator selects that environment directly; otherwise, the generator is first configured for a new language and architecture.

Because of its open architecture, the generator can be configured to reside on any new architecture (or interface to any outside environment), e.g., to a language, communications package, an internet interface, a database package, or an operating system of choice; or it can be configured to interface to the users own legacy code. Once configured for a new environment, an existing system can be automatically regenerated to reside on that new environment. This open architecture approach, which lends itself to true component based development, provides more flexibility to the user when changing requirements or architectures; or when moving from an older technology to a newer one.

It then becomes possible to execute the resulting system. If it is software, the system can undergo testing for further user intent errors. It becomes operational after testing. Application changes are made to the requirements definition—not to the code. Target architecture changes are made to the configuration of the generator environment (which generates one of a possible set of implementations from the model)—not to the code. If the real system is hardware or peopleware, the software system serves as a simulation upon which the real system can be based. Once a system has been developed, the system and the process used to develop it are analyzed to understand how to improve the next round of system development.

Seamless integration is provided throughout from systems to software, requirements to design to code to tests to other requirements and back again; level to level, and layer to layer. The developer is able to trace from requirements to code and back again.

Given an automation that has these capabilities, it should be no surprise that it has been defined with itself and that it continues to automatically generate itself as it evolves with changing architectures and changing technologies. Table 17.2 contains a summary of some of the differences between the more modern preventative paradigm and the traditional approach.

A relatively small set of things is needed to master the concepts behind DBTF. Everything else can be derived, leading to powerful reuse capabilities for building systems. It quickly becomes clear why it is no longer necessary to add features to the language or changes to a developed application in an ad hoc fashion, since each new aspect is ultimately and inherently derived from its mathematical foundations.

Although this approach addresses many of the challenges and solves many of the problems of traditional software environments, it could take time before this paradigm (or one with similar properties) is adopted by the more mainstream users, since it requires a change to the corporate culture. The same has been true in related fields. At the time when the computer was first invented and manual calculators were used for almost every computation; it has been said that it was believed by hardware pioneers like Ken Olson, founder of Digital Equipment Corporation, that there would only be a need for four computers in the world. It took awhile for the idea of computers and what they were capable of to catch on. Such could be the case for a more advanced software paradigm as well. That is, it could take awhile for software to be truly something to manufacture as opposed to being handcrafted as it is in today's traditional development environments.

© 2006 by Taylor & Francis Group, LLC

TABLE 17.2 A Comparison

Traditional (After the Fact)	DBTF (Development Before the Fact)
Integration ad hoc, if at all	*Integration*
Mismatched methods, objects, phases, products, architectures, applications and environment	Seamless life cycle: methods, objects, phases, products, architectures, applications and environment
System not integrated with software	System integrated with software
Function oriented *or* object oriented	System oriented objects: integration of function, timing, and object oriented
GUI not integrated with application	GUI integrated with application
Simulation not integrated with software code	Simulation integrated with software code
Behavior uncertain until after delivery	*Correctness by built-in language properties*
Interface errors abound and infiltrate the system (over 75% of all errors)	*No interface errors*
Most of those found are found after implementation	All found before implementation
Some found manually	All found by automatic and static analysis
Some found by dynamic runs analysis	
Some never found	Always found
Ambiguous requirements, specifications, designs . . . introduce chaos, confusion and complexity	*Unambiguous requirements, specifications, designs . . . remove chaos, confusion and complexity*
Informal or semi-formal language	Formal, but friendly language
Different phases, languages and tools	All phases, same language and tools
Different language for other systems than for software	Same language for software, hardware and any other system
No guarantee of function integrity after implementation	*Guarantee of function integrity after implementation*
Inflexible: Systems not traceable or evolvable	*Flexible: Systems traceable and evolvable*
Locked in bugs, requirements products, architectures, etc.	Open architecture
Painful transition from legacy	Smooth transition from legacy
Maintenance performed at code level	Maintenance performed at spec level
Reuse not inherent	*Inherent Reuse*
Reuse is adhoc	Every object a candidate for reuse
Customization and reuse are mutually exclusive	Customization increases the reuse pool
Automation supports manual process instead of doing real work	*Automation does real work*
Mostly manual: documentation, programming, test generation, traceability, integration	Automatic programming, documentation, test generation, traceability, integration
Limited, incomplete, fragmented, disparate and inefficient	100% code automatically generated for any kind of software
Product x not defined and developed with itself	*001 defined with and generated by itself*
	#1 in all evaluations
Dollars wasted, error prone systems	*Ultra-reliable systems with unprecedented productivity in their development*
High risk	Low risk
Not cost effective	10 to 1, 20 to 1, 50 to 1 . . . dollars saved/dollars made
Difficult to meet schedules	Minimum time to complete
Less of what you need and more of what you do not need	No more, no less of what you need

17.9 Select the Right Paradigm and then Automate

Where software engineering fails is in its inability to grasp that not only the right paradigm (out of many paradigms) must be selected, but that the paradigm must be part of an environment that provides an inherent and integrated automated means to solve the problem at hand. What this means is that the paradigm must be coupled with an integrated set of tools with which to implement the results of utilizing

© 2006 by Taylor & Francis Group, LLC

the paradigm to design and develop the system. Essentially, the paradigm generates the model and a toolset must be provided to generate the system.

Businesses that expected a big productivity payoff from investing in technology are, in many cases, still waiting to collect. A substantial part of the problem stems from the manner in which organizations are building their automated systems. Although hardware capabilities have increased dramatically, organizations are still mired in the same methodologies that saw the rise of the behemoth mainframes. Obsolete methodologies simply cannot build new systems.

There are other changes as well. Users demand much more functionality and flexibility in their systems. And given the nature of many of the problems to be solved by this new technology, these systems must also be error free.

Where the biological superorganism has built-in control mechanisms fostering quality and productivity, up until now, the silicon superorganism has had none. Hence, the productivity paradox. Often, the only way to solve major issues or to survive tough times is through nontraditional paths or innovation. One must create new methods or new environments for using new methods. Innovation for success often starts with a look at mistakes from traditional systems. The first step is to recognize the true root problems, then categorize them according to how they might be prevented. Derivation of practical solutions is a logical next step. Iterations of the process entail looking for new problem areas in terms of the new solution environment and repeating the scenario. That is how DBTF came into being.

With DBTF, all aspects of system design and development are integrated with one systems language and its associated automation. Reuse naturally takes place throughout the life cycle. Objects, no matter how complex, can be reused and integrated. Environment configurations for different kinds of architectures can be reused. A newly developed system can be safely reused to increase even further the productivity of the systems developed with it.

This preventative approach can support a user in addressing many of the challenges presented in today's software development environments. There will, however, always be more to do to capitalize on this technology. That is part of what makes a technology like this so interesting to work with. Because it is based on a different premise or set of assumptions (axioms), a significant number of things can and will change because of it. There is the continuing opportunity for new research projects and new products. Some problems can be solved, because of the language, that could not be solved before. Software development, as we have known it, will never be the same. Many things will no longer need to exist—they, in fact, will be rendered extinct, just as that phenomenon occurs with the process of natural selection in the biological system. Techniques for bridging the gap from one phase of the life cycle to another become obsolete. Techniques for maintaining the source code as a separate process are no longer needed, since the source is automatically generated from the requirements specification. Verification, too, becomes obsolete. Techniques for managing paper documents give way to entering requirements and their changes directly into the requirements specification database. Testing procedures and tools for finding most errors are no longer needed because those errors no longer exist. Tools to support programming as a manual process are no longer needed.

Compared to development using traditional techniques, the productivity of DBTF developed systems has been shown to be significantly greater (DOD, 1992; Krut, 1993; Ouyang, 1995; Keyes, 2000). Upon further analysis, it was discovered that the productivity was greater the larger and more complex the system—the opposite of what one finds with traditional systems development. This is, in pa2004rt, because of the high degree of DBTF's support of reuse. The larger a system, the more it has the opportunity to capitalize on reuse. As more reuse is employed, productivity *continues* to increase. Measuring productivity becomes a process of relativity—that is, relative to the last system developed.

Capitalizing on reusables within a DBTF environment is an ongoing area of research interest[1] (Hamilton, 2003–2004). An example is understanding the relationship between types of reusables and metrics.

[1] An initial investigation of the form of a common software interface for use with common guidance systems under contract to ARES Corporation as a part of its contract with TACOM ARDEC, Mr. Nigel Gray, program manager. Mr.Gray is the originator of the Common Guidance Common Sense (CGCS) and Deeply Integrated Guidance and

© 2006 by Taylor & Francis Group, LLC

This takes into consideration that a reusable can be categorized in many ways. One is according to the manner in which its use saves time (which translates to how it impacts cost and schedules). More intelligent tradeoffs can then be made. The more we know about how some kinds of reusables are used, the more information we have to estimate costs for an overall system. Keep in mind, also, that the traditional methods for estimating time and costs for developing software are no longer valid for estimating systems developed with preventative techniques.

There are other reasons for this higher productivity as well, such as the savings realized and time saved due to tasks and processes that are no longer necessary with the use of this preventative approach. There is less to learn and less to do—less analysis, little or no implementation, less testing, less to manage, less to document, less to maintain, and less to integrate. This is because a major part of these areas has been automated or because of what inherently takes place because of the nature of the formal systems language.

The paradigm shift occurs once a designer realizes that many of the old tools are no longer needed to design and develop a system. For example, with one formal semantic language to define and integrate all aspects of a system, diverse modeling languages (and methodologies for using them), each of which defines only part of a system, are no longer necessary. There is no longer a need to reconcile multiple techniques with semantics that interfere with each other.

Software is a relatively young technological field that is still in a constant state of change. Changing from a traditional software environment to a preventative one is like going from the typewriter to the word processor. Whenever there is any major change, there is always the initial overhead needed for learning the new way of doing things. But, as with the word processor, progress begets progress.

In the end, it is the combination of the methodology and the technology that executes it that forms the foundation of successful software. Software is so ingrained in our society that its success or failure will dramatically influence both the operation and the success of businesses and government. For that reason, today's decisions about systems engineering and software development will have far reaching effects.

Collective experience strongly confirms that quality and productivity increase with the increased use of properties of preventative systems. In contrast to the "better late than never" after the fact philosophy, the preventative philosophy is to solve—or possible, prevent—a given problem as early as possible. This means finding a problem statically is better than finding it dynamically. Preventing it by the way a system is defined is even better. Better yet, is not having to define (and build) it at all.

Reusing a reliable system is better than reusing one that is not reliable. Automated reuse is better than manual reuse. Inherent reuse is better than automated reuse. Reuse that can evolve is better than one that cannot evolve. Best of all is reuse that approaches wisdom itself. Then, have the wisdom to use it. One such wisdom is that of applying a common systems paradigm to systems and software, unifying their understanding with a commonly held set of system semantics.

DBTF embodies a preventative systems approach, the language supports its representation, and its automation supports its application and use. Each is evolutionary (in fact, recursively so), with experience feeding the theory, the theory feeding the language, which in turn feeds the automation. All are used, in concert, to design systems and build software. The answer continues to be in the results just as in the biological system and the goal is that the systems of tomorrow will inherit the best of the systems of today.

Defining Terms

Data base management system (DBMS): The computer program that is used to control and provide rapid access to a database. A language is used with the DBMS to control the functions that a DBMS provides. For example, SQL is the language that is used to control all of the functions that a relational architecture based DBMS provides for its users, including data definition, data retrieval, data manipulation, access control, data sharing, and data integrity.

Navigation "smart munitions" and other military and aerospace system applications. The common software interface for these systems will provide a common means for user programming of the internal mission processor in these systems, independent of the embedded operating system (HTI, 2003–2004)

© 2006 by Taylor & Francis Group, LLC

Graphical user interface (GUI): The ultimate user interface, by which the deployed system interfaces with the computer most productively, using visual means. Graphical user interfaces provide a series of intuitive, colorful, and graphical mechanisms which enable the end-user to view, update, and manipulate information.

Formal system: A system defined in terms of a known set of axioms (or assumptions); it is therefore mathematically based (for example, a DBTF system is based on a set of axioms of control). Some of its properties are that it is consistent and logically complete. A system is consistent if it can be shown that no assumption of the system contradicts any other assumption of that system. A system is logically complete if the assumptions of the method completely define a given set of properties. This assures that a model of the method has that set of properties. Other properties of the models defined with the method may not be provable from the method's assumptions. A logically complete system has a semantic basis (i.e., a way of expressing the meaning of that system's objects). In terms of the semantics of a DBTF system, this means it has no interface errors and is unambiguous, contains what is necessary and sufficient, and has a unique state identification.

Interface: A seam between objects, or programs, or systems. It is at this juncture that many errors surface. Software can interface with hardware, humans, and other software.

Methodology: A set of procedures, precepts, and constructs for the construction of software.

Metrics: A series of formulas which measure such things as quality and productivity.

Operating system: A program that manages the application programs that run under its control.

Software architecture: The structure and relationships among the components of software.

Acknowledgment

The author is indebted to Jessica Keyes for her helpful suggestions.

References

Boehm, B.W. 1981. *Software Engineering Economics.* Prentice-Hall, Englewood Cliffs, NJ.

Booch, G., Rumbaugh, J., and Jacobson, I. 1999. *The Unified Modeling Language User Guide.* Addison-Wesley, Reading, MA.

Department of Defense. 1992. *Software engineering tools experiment-Final report*. Vol. 1, Experiment Summary, Table 1, p. 9. Strategic Defense Initiative, Washington, DC.

Goldberg, A. and Robson, D. 1983. *Smalltalk-80 the Language and its Implementation.* Addison-Wesley, Reading, MA.

Gosling, J., Joy, B., and Steele, G. 1996. *The Java Language Specification*, Addison-Wesley, Reading, MA.

Hamilton, M. 1994. Development before the fact in action. *Electronic Design*, June 13. ES.

Hamilton, M. 1994. Inside development before the fact. *Electronic Design*, April 4. ES.

Hamilton, M. (In press). *System Oriented Objects: Development Before the Fact*.

Hamilton, M. and Hackler, W.R. 2000. Towards Cost Effective and Timely End-to-End Testing, *HTI*, prepared for Army Research Laboratory, Contract No. DAKF11-99-P-1236.

HTI Technical Report, *Common Software Interface for Common Guidance Systems*, US. Army DAAE30-02-D-1020 and DAAB07-98-D-H502/0180, Precision Guided Munitions, Under the direction of Cliff McLain, ARES, and Nigel Gray, COTR ARDEC, Picatinny Arsenal, NJ, 2003–2004.

Harbison, S.P. and Steele, G.L., Jr. 1997. *C A Reference Manual*. Prentice-Hall, Englewood Cliffs, NJ.

Jones, T.C. 1977. Program quality and programmer productivity. *IBM Tech.* Report TR02.764, Jan.: 80, Santa Teresa Labs., San Jose, CA.

Keyes, J. The Ultimate Internet Developers Sourcebook, Chapter 42, *Developing Web Applications with 001*, AMACOM.

Keyes, J. 2000. *Internet Management*, chapters 30–33, on 001-developed systems for the Internet, Auerbach.

Krut, B. Jr. 1993. *Integrating 001 Tool Support in the Feature-Oriented Domain Analysis Methodology* (CMU/SEI-93-TR-11, ESC-TR-93-188). Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University.

© 2006 by Taylor & Francis Group, LLC

Lickly, D. J. 1974. *HAL/S Language Specification,* Intermetrics, prepared for NASA Lyndon Johnson Space Center.

Lientz, B.P. and Swanson, E.B. 1980. *Software Maintenance Management*. Addison-Wesley, Reading, MA.

Martin, J. and Finkelstein, C.B. 1981. *Information Engineering.* Savant Inst., Carnforth, Lancashire, UK.

Meyer, B. 1992. *Eiffel the Language.* Prentice-Hall, New York.

NSQE Experiment: http://hometown.aol.com/ONeillDon/index.html, Director of Defense Research and Engineering (DOD Software Technology Strategy), 1992–2002.

Ouyang, M., Golay, M. W. 1995. *An Integrated Formal Approach for Developing High Quality Software of Safety-Critical Systems*, Massachusetts Institute of Technology, Cambridge, MA, Report No. MIT-ANP-TR-035.

Software Engineering Inst. 1991. *Capability Maturity Model.* Carnegie-Mellon University Pittsburgh, PA.

Stefik, M. and Bobrow, D.J. 1985. Object-Oriented Programming: Themes and Variations. *AI Magazine*, Zerox Corporation, pp. 40–62.

Stroustrup, B. 1994. *The Design and Evolution of C++*. AT&T Bell Laboratories, Murray Hill, NJ.

Stroustrup, B. 1997. *The C++ Programming Language*, Addison-Wesley, Reading, MA.

Yourdon E. and Constantine, L. 1978. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design,* Yourdan Press, New York.

Further Information

Hamilton, M. and Hackler, R. 1990. 001: A rapid development approach for rapid prototyping based on a system that supports its own life cycle. *IEEE Proceedings*, First International Workshop on Rapid System Prototyping (Research Triangle Park, NC) pp. 46–62, June 4.

Hamilton, M. 1986. Zero-defect software: The elusive goal. *IEEE Spectrum* 23(3):48–53, March.

Hamilton, M. and Zeldin, S. 1976. Higher Order Software—A Methodology for Defining Software. *IEEE Transactions on Software Engineering*, vol. SE-2, no. 1.

McCauley, B. 1993. Software Development Tools in the 1990s. AIS Security Technology for Space Operations Conference, Houston, Texas.

Schindler, M. 1990. *Computer Aided Software Design.* John Wiley & Sons, New York.

The 001 *Tool Suite Reference Manual*, Version 3, Jan. 1993. Hamilton Technologies Inc., Cambridge, MA.

© 2006 by Taylor & Francis Group, LLC

18

Neural Networks and Fuzzy Systems

18.1 Neural Networks and Fuzzy Systems **18**-1
18.2 Neuron cell **18**-1
18.3 Feedforward Neural Networks **18**-3
18.4 Learning Algorithms for Neural Networks **18**-5
Hebbian Learning Rule • Correlation Learning Rule • Instar Learning Rule • Winner Takes All (WTA) • Outstar Learning Rule • Widrow-Hoff LMS Learning Rule • Linear Regression • Delta Learning Rule • Error Backpropagation Learning
18.5 Special Feedforward Networks **18**-11
Functional Link Network • Cascade Correlation Architecture
18.6 Recurrent Neural Networks **18**-16
Hopfield Network • Autoassociative Memory • Bidirectional Associative Memories (BAM)
18.7 Fuzzy Systems **18**-18
Fuzzification • Rule Evaluation • Defuzzification
18.8 Design Example **18**-21
18.9 Genetic Algorithms **18**-21
Coding and Initialization • Selection and Reproduction • Reproduction • Mutation

Bogdan M. Wilamowski

18.1 Neural Networks and Fuzzy Systems

New and better electronic devices have inspired researchers to build intelligent machines operating in a fashion similar to the human nervous system. Fascination with this goal started when McCulloch and Pitts (1943) developed their model of an elementary computing neuron and when Hebb (1949) introduced his *learning rules*. A decade latter Rosenblatt (1958) introduced the **perceptron** concept. In the early 1960s Widrow and Holf (1960, 1962) developed intelligent systems such as ADALINE and MADALINE. Nillson (1965) in his book *Learning Machines* summarized many developments of that time. The publication of the Mynsky and Paper (1969) book, with some discouraging results, stopped for sometime the fascination with artificial neural networks, and achievements in the mathematical foundation of the **backpropagation** algorithm by Werbos (1974) went unnoticed. The current rapid growth in the area of neural networks started with the Hopfield (1982, 1984) recurrent network, Kohonen (1982) unsupervised training algorithms, and a description of the backpropagation algorithm by Rumelhart et al. (1986).

18.2 Neuron cell

A biological neuron is a complicated structure, which receives trains of pulses on hundreds of *excitatory* and *inhibitory* inputs. Those incoming pulses are summed with different weights (averaged) during the time period of *latent summation*. If the summed value is higher than a threshold, then the neuron itself is

© 2006 by Taylor & Francis Group, LLC

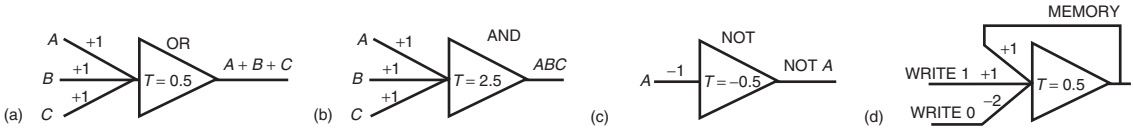


FIGURE 18.1 OR, AND, NOT, and MEMORY operations using networks with McCulloch–Pitts neuron model.

generating a pulse, which is sent to neighboring neurons. Because incoming pulses are summed with time, the neuron generates a pulse train with a higher frequency for higher positive excitation. In other words, if the value of the summed weighted inputs is higher, the neuron generates pulses more frequently. At the same time, each neuron is characterized by the nonexcitability for a certain time after the firing pulse. This so-called *refractory period* can be more accurately described as a phenomenon where after excitation the threshold value increases to a very high value and then decreases gradually with a certain time constant. The refractory period sets soft upper limits on the frequency of the output pulse train. In the biological neuron, information is sent in the form of frequency modulated pulse trains.

This description of neuron action leads to a very complex neuron model, which is not practical. McCulloch and Pitts (1943) show that even with a very simple neuron model, it is possible to build logic and memory circuits. Furthermore, these simple neurons with thresholds are usually more powerful than typical logic gates used in computers. The McCulloch-Pitts neuron model assumes that incoming and outgoing signals may have only binary values 0 and 1. If incoming signals summed through positive or negative weights have a value larger than threshold, then the neuron output is set to 1. Otherwise, it is set to 0.

$$T = \begin{cases} 1 & \text{if} \quad net \geq T \\ 0 & \text{if} \quad net < T \end{cases} \tag{18.1}$$

where T is the threshold and *net* value is the weighted sum of all incoming signals

$$net = \sum_{i=1}^{n} w_i x_i \tag{18.2}$$

Examples of McCulloch-Pitts neurons realizing OR, AND, NOT, and MEMORY operations are shown in Fig. 18.1. Note that the structure of OR and AND gates can be identical. With the same structure, other logic functions can be realized, as Fig. 18.2 shows.

The perceptron model has a similar structure. Its input signals, the weights, and the thresholds could have any positive or negative values. Usually, instead of using variable threshold, one additional constant input with a negative or positive weight can added to each neuron, as Fig. 18.3 shows. In this case, the threshold is always set to be zero and the net value is calculated as

$$net = \sum_{i=1}^{n} w_i x_i + w_{n+1} \tag{18.3}$$

where w_{n+1} has the same value as the required threshold and the opposite sign. Single-layer perceptrons were successfully used to solve many pattern classification problems. The hard threshold activation functions are given by

$$o = f(net) = \frac{\text{sgn}(net) + 1}{2} = \begin{cases} 1 & \text{if} \quad net \geq 0 \\ 0 & \text{if} \quad net < 0 \end{cases} \tag{18.4}$$

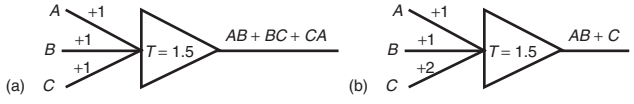


FIGURE 18.2 Other logic functions realized with McCulloch–Pitts neuron model.

© 2006 by Taylor & Francis Group, LLC

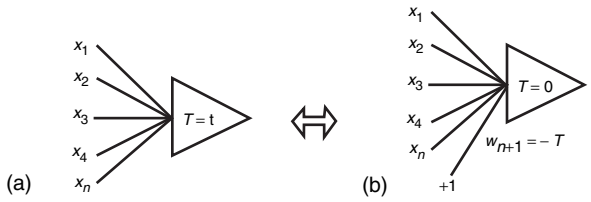


FIGURE 18.3 Threshold implementation with an additional weight and constant input with +1 value: (a) Neuron with threshold T, (b) modified neuron with threshold $T = 0$ and additional weight equal to $-T$.

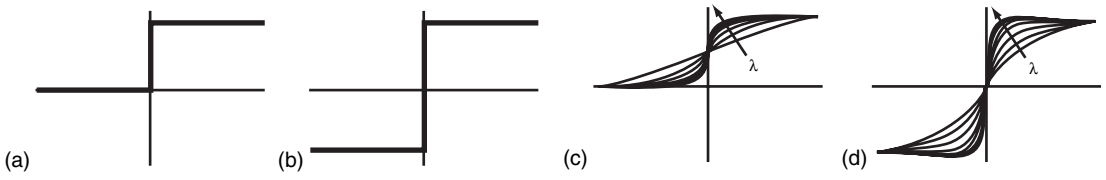


FIGURE 18.4 Typical activation functions: (a) hard threshold unipolar, (b) hard threshold bipolar, (c) continuous unipolar, (d) continuous bipolar.

for **unipolar** neurons and

$$o = f(net) = \text{sgn}(net) = \begin{cases} 1 & \text{if} \quad net \geq 0 \\ -1 & \text{if} \quad net < 0 \end{cases} \tag{18.5}$$

for **bipolar** neurons. For these types of neurons, most of the known training algorithms are able to adjust weights only in single-layer networks.

Multilayer neural networks usually use continuous activation functions, either unipolar

$$o = f(net) = \frac{1}{1 + \exp(-\lambda net)} \tag{18.6}$$

or bipolar

$$o = f(net) = \tanh(0.5\lambda net) = \frac{2}{1 + \exp(-\lambda net)} - 1 \tag{18.7}$$

These continuous activation functions allow for the gradient-based training of multilayer networks. Typical activation functions are shown in Fig. 18.4. In the case when neurons with additional threshold input are used (Fig. 18.3(b)), the λ parameter can be eliminated from Eqs. (18.6) and (18.7) and the steepness of the neuron response can be controlled by the weight scaling only. Therefore, there is no real need to use neurons with variable gains.

Note, that even neuron models with continuous activation functions are far from an actual biological neuron, which operates with frequency modulated pulse trains.

18.3 Feedforward Neural Networks

Feedforward neural networks allow only one directional signal flow. Furthermore, most feedforward neural networks are organized in layers. An example of the three-layer feedforward neural network is shown in Fig. 18.5. This network consists of input nodes, two *hidden layers*, and an output layer.

A single neuron is capable of separating input patterns into two categories, and this separation is linear. For example, for the patterns shown in Fig. 18.6, the separation line is crossing x_1 and x_2 axes at points x_{10} and x_{20}. This separation can be achieved with a neuron having the following weights: $w_1 = 1/x_{10}$,

© 2006 by Taylor & Francis Group, LLC

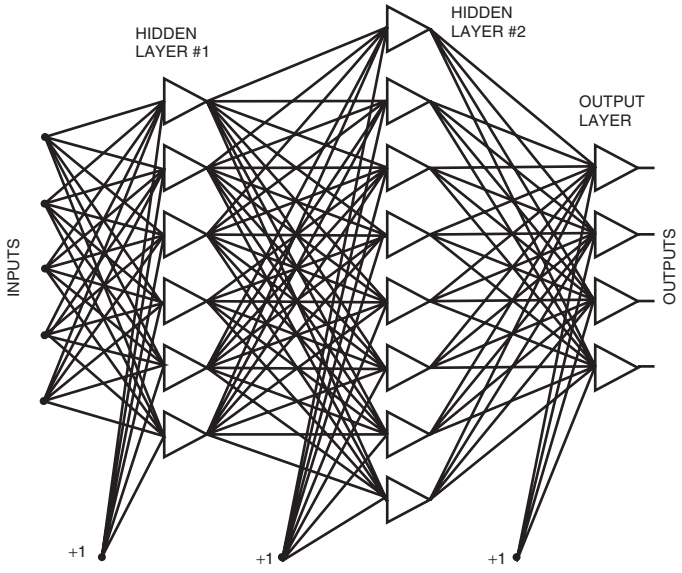


FIGURE 18.5 An example of the three layer feedforward neural network, which is sometimes known also as the backpropagation network.

$w_2 = 1/x_{20}$, and $w_3 = -1$. In general for n dimensions, the weights are

$$w_i = \frac{1}{x_{i0}} \quad \text{for} \quad w_{n+1} = -1 \tag{18.8}$$

One neuron can divide only linearly separated patterns. To select just one region in n-dimensional input space, more than $n + 1$ neurons should be used. If more input clusters are to be selected, then the number of neurons in the input (hidden) layer should be properly multiplied. If the number of neurons in the input (hidden) layer is not limited, then all classification problems can be solved using the three-layer network. An example of such a neural network, classifying three clusters in the two-dimensional space, is shown in Fig. 18.7. Neurons in the first hidden layer create the separation lines between input clusters. Neurons in the second hidden layer perform the AND operation, as shown in Fig. 18.1(b). Output neurons perform the OR operation as shown in Fig. 18.1(a), for each category. The linear separation property of neurons makes some problems specially difficult for neural networks, such as exclusive OR, parity computation for several bits, or to separate patterns laying on two neighboring spirals.

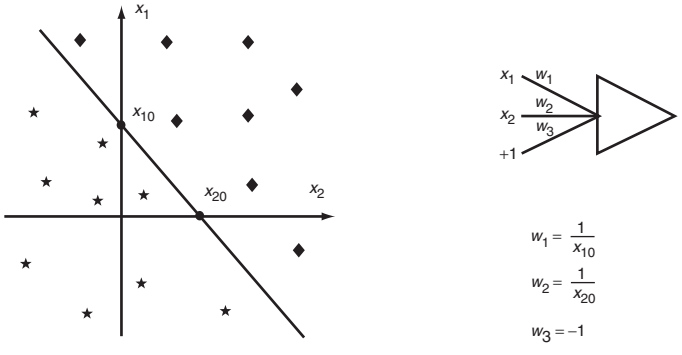


FIGURE 18.6 Illustration of the property of linear separation of patterns in the two-dimensional space by a single neuron.

© 2006 by Taylor & Francis Group, LLC

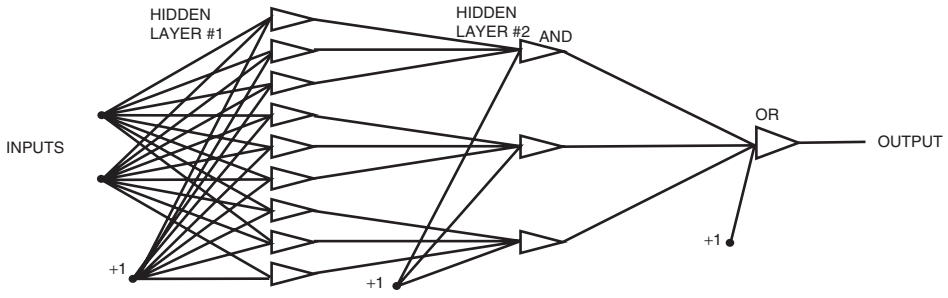


FIGURE 18.7 An example of the three layer neural network with two inputs for classification of three different clusters into one category. This network can be generalized and can be used for solution of all classification problems.

The feedforward neural network is also used for nonlinear transformation (mapping) of a multidimensional input variable into another multidimensional variable in the output. In theory, any input-output mapping should be possible if the neural network has enough neurons in hidden layers (size of output layer is set by the number of outputs required). In practice, this is not an easy task. Presently, there is no satisfactory method to define how many neurons should be used in hidden layers. Usually, this is found by the trial-and-error method. In general, it is known that if more neurons are used, more complicated shapes can be mapped. On the other hand, networks with large numbers of neurons lose their ability for generalization, and it is more likely that such networks will also try to map noise supplied to the input.

18.4 Learning Algorithms for Neural Networks

Similarly to the biological neurons, the weights in artificial neurons are adjusted during a training procedure. Various learning algorithms were developed, and only a few are suitable for multilayer neuron networks. Some use only local signals in the neurons, others require information from outputs; some require a supervisor who knows what outputs should be for the given patterns, and other unsupervised algorithms need no such information. Common learning rules are described in the following sections.

Hebbian Learning Rule

The Hebb (1949) learning rule is based on the assumption that if two neighbor neurons must be activated and deactivated at the same time, then the weight connecting these neurons should increase. For neurons operating in the opposite phase, the weight between them should decrease. If there is no signal correlation, the weight should remain unchanged. This assumption can be described by the formula

$$\Delta w_{ij} = c x_i o_j \tag{18.9}$$

where

w_{ij} = weight from ith to jth neuron
c = learning constant
x_i = signal on the ith input
o_j = output signal

The training process starts usually with values of all weights set to zero. This learning rule can be used for both soft and hard threshold neurons. Since desired responses of neurons is not used in the learning procedure, this is the **unsupervised learning** rule. The absolute values of the weights are usually proportional to the learning time, which is undesired.

Correlation Learning Rule

The correlation learning rule is based on a similar principle as the Hebbian learning rule. It assumes that weights between simultaneously responding neurons should be largely positive, and weights between neurons with opposite reaction should be largely negative.

© 2006 by Taylor & Francis Group, LLC

Contrary to the Hebbian rule, the correlation rule is the **supervised learning**. Instead of actual response o_j, the desired response d_j is used for the weight change calculation

$$\Delta w_{ij} = c x_i d_j \tag{18.10}$$

This training algorithm usually starts with initialization of weights to zero values.

Instar Learning Rule

If input vectors and weights are normalized, or they have only binary bipolar values (-1 or $+1$), then the *net* value will have the largest positive value when the weights and the input signals are the same. Therefore, weights should be changed only if they are different from the signals

$$\Delta w_i = c(x_i - w_i) \tag{18.11}$$

Note, that the information required for the weight is taken only from the input signals. This is a very local and unsupervised learning algorithm.

Winner Takes All (WTA)

The WTA is a modification of the instar algorithm where weights are modified only for the neuron with the highest *net* value. Weights of remaining neurons are left unchanged. Sometimes this algorithm is modified in such a way that a few neurons with the highest net values are modified at the same time. Although this is an unsupervised algorithm because we do not know what are desired outputs, there is a need for "judge" or "supervisor" to find a winner with a largest net value. The WTA algorithm, developed by Kohonen (1982), is often used for automatic clustering and for extracting statistical properties of input data.

Outstar Learning Rule

In the outstar learning rule, it is required that weights connected to a certain node should be equal to the desired outputs for the neurons connected through those weights

$$\Delta w_{ij} = c(d_j - w_{ij}) \tag{18.12}$$

where d_j is the desired neuron output and c is small learning constant, which further decreases during the learning procedure. This is the supervised training procedure because desired outputs must be known. Both instar and outstar learning rules were developed by Grossberg (1969).

Widrow-Hoff LMS Learning Rule

Widrow and Hoff (1960, 1962) developed a supervised training algorithm which allows training a neuron for the desired response. This rule was derived so that the square of the difference between the net and output value is minimized.

$$Error_j = \sum_{p=1}^{P} (net_{jp} - d_{jp})^2 \tag{18.13}$$

where:
$Error_j$ = error for jth neuron
P = number of applied patterns
d_{jp} = desired output for jth neuron when pth pattern is applied
net = given by Eq. (18.2).

© 2006 by Taylor & Francis Group, LLC

This rule is also known as the least mean square (LMS) rule. By calculating a derivative of Eq. (18.13) with respect to w_{ij}, a formula for the weight change can be found,

$$\Delta w_{ij} = c x_i \sum_{p=1}^{P} (d_{jp} - net_{jp}) \tag{18.14}$$

Note that weight change Δw_{ij} is a sum of the changes from each of the individual applied patterns. Therefore, it is possible to correct the weight after each individual pattern was applied. This process is known as *incremental updating*; *cumulative updating* is when weights are changed after all patterns have been applied. Incremental updating usually leads to a solution faster, but it is sensitive to the order in which patterns are applied. If the learning constant c is chosen to be small, then both methods give the same result. The LMS rule works well for all types of activation functions. This rule tries to enforce the *net* value to be equal to desired value. Sometimes this is not what the observer is looking for. It is usually not important what the *net* value is, but it is important if the *net* value is positive or negative. For example, a very large *net* value with a proper sign will result in correct output and in large error as defined by Eq. (18.13) and this may be the preferred solution.

Linear Regression

The LMS learning rule requires hundreds or thousands of iterations, using formula (18.14), before it converges to the proper solution. Using the linear regression rule, the same result can be obtained in only one step.

Considering one neuron and using vector notation for a set of the input patterns $\boldsymbol{X}$ applied through weight vector $\boldsymbol{w}$, the vector of *net* values $\boldsymbol{net}$ is calculated using

$$\boldsymbol{X}w = \boldsymbol{net} \tag{18.15}$$

where

$\boldsymbol{X}$ = rectangular array $(n+1) \times p$
n = number of inputs
p = number of patterns

Note that the size of the input patterns is always augmented by one, and this additional weight is responsible for the threshold (see Fig. 18.3(b)). This method, similar to the LMS rule, assumes a linear activation function, and so the *net* values *net* should be equal to desired output values $\boldsymbol{d}$

$$\boldsymbol{X}\boldsymbol{w} = \boldsymbol{d} \tag{18.16}$$

Usually $p > n + 1$, and the preceding equation can be solved only in the least mean square error sense. Using the vector arithmetic, the solution is given by

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{d} \tag{18.17}$$

When traditional method is used the set of p equations with $n + 1$ unknowns Eq. (18.16) has to be converted to the set of $n + 1$ equations with $n + 1$ unknowns

$$\boldsymbol{Y}\boldsymbol{w} = \boldsymbol{z} \tag{18.18}$$

where elements of the $\boldsymbol{Y}$ matrix and the $\boldsymbol{z}$ vector are given by

$$y_{ij} = \sum_{p=1}^{P} x_{ip} x_{jp} \qquad z_i = \sum_{p=1}^{P} x_{ip} d_p \tag{18.19}$$

Weights are given by Eq. (18.17) or they can be obtained by a solution of Eq. (18.18).

© 2006 by Taylor & Francis Group, LLC

Delta Learning Rule

The LMS method assumes linear activation function ***net*** $= \boldsymbol{o}$, and the obtained solution is sometimes far from optimum, as is shown in Fig. 18.8 for a simple two-dimensional case, with four patterns belonging to two categories. In the solution obtained using the LMS algorithm, one pattern is misclassified. If error is defined as

$$Error_j = \sum_{p=1}^{P} (o_{jp} - d_{jp})^2 \tag{18.20}$$

Then the derivative of the error with respect to the weight w_{ij} is

$$\frac{d\,Error_j}{dw_{ij}} = 2\sum_{p=1}^{P} (o_{jp} - d_{jp}) \frac{df(net_{jp})}{d\,net_{jp}} x_i \tag{18.21}$$

since $o = f(net)$ and the *net* is given by Eq. (18.2). Note that this derivative is proportional to the derivative of the activation function $f'(net)$. Thus, this type of approach is possible only for continuous activation functions and this method cannot be used with hard activation functions (18.4) and (18.5). In this respect the LMS method is more general. The derivatives most common continuous activation functions are

$$f' = o(1 - o) \tag{18.22}$$

for the unipolar (Eq. (18.6)) and

$$f' = 0.5(1 - o^2) \tag{18.23}$$

for the bipolar (Eq. (18.7)).

Using the cumulative approach, the neuron weight w_{ij} should be changed with a direction of gradient

$$\Delta w_{ij} = c\, x_i \sum_{p=1}^{P} (d_{jp} - o_{jp})\, f'_{jp} \tag{18.24}$$

in case of the incremental training for each applied pattern

$$\Delta w_{ij} = c x_i f'_j (d_j - o_j) \tag{18.25}$$

the weight change should be proportional to input signal x_i, to the difference between desired and actual outputs $d_{jp} - o_{jp}$, and to the derivative of the activation function f'_{jp}. Similar to the LMS rule, weights can be updated in both the incremental and the cumulative methods. In comparison to the LMS rule, the delta rule always leads to a solution close to the optimum. As it is illustrated in Fig. 18.8, when the delta rule is used, all four patterns are classified correctly.

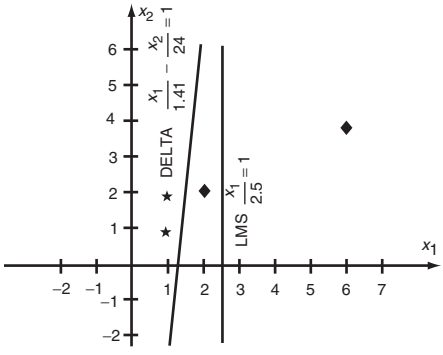


FIGURE 18.8 An example with a comparison of results obtained using LMS and delta training algorithms. Note that LMS is not able to find the proper solution.

© 2006 by Taylor & Francis Group, LLC

Error Backpropagation Learning

The delta learning rule can be generalized for multilayer networks. Using an approach similiar to the delta rule, the gradient of the global error can be computed with respect to each weight in the network. Interestingly,

$$\Delta w_{ij} = c x_i f_j' E_j \tag{18.26}$$

where

c = learning constant
x_i = signal on the ith neuron input
f_j' = derivative of activation function

The cumulative error E_j on neuron output is given by

$$E_j = \frac{1}{f_j'} \sum_{k=1}^{K} (o_k - d_k) A_{jk} \tag{18.27}$$

where K is the number of network outputs A_{jk} is the small signal gain from the input of jth neuron to the kth network output, as Fig. 18.9 shows. The calculation of the backpropagating error starts at the output layer and cumulative errors are calculated layer by layer to the input layer. This approach is not practical from the point of view of hardware realization. Instead, it is simpler to find signal gains from the input of the jth neuron to each of the network outputs (Fig. 18.9). In this case, weights are corrected using

$$\Delta w_{ij} = c x_i \sum_{k=1}^{K} (o_k - d_k) A_{jk} \tag{18.28}$$

Note that this formula is general, regardless of if neurons are arranged in layers or not. One way to find gains A_{jk} is to introduce an incremental change on the input of the jth neuron and observe the change in the kth network output. This procedure requires only forward signal propagation, and it is easy to implement in a hardware realization. Another possible way is to calculate gains through each layer and then find the total gains as products of layer gains. This procedure is equally or less computationally intensive than a calculation of cumulative errors in the error backpropagation algorithm.

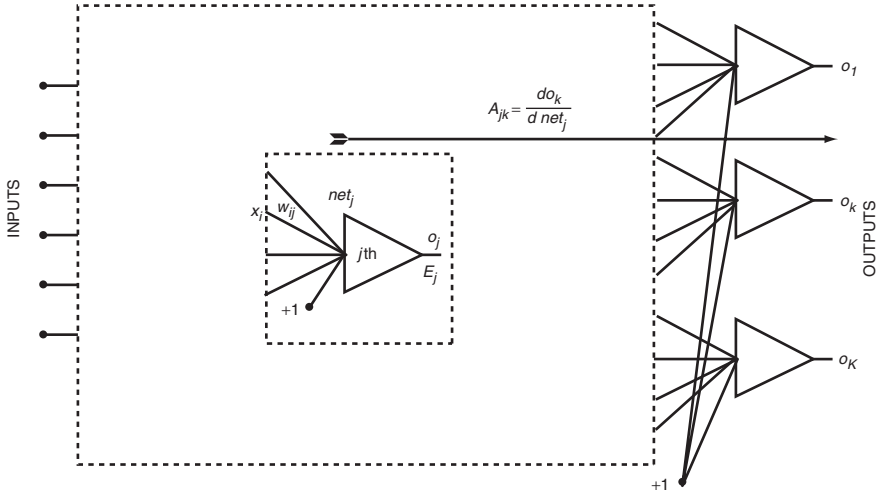


FIGURE 18.9 Illustration of the concept of gain computation in neural networks.

© 2006 by Taylor & Francis Group, LLC

The backpropagation algorithm has a tendency for oscillation. To smooth the process, the weights increment Δw_{ij} can be modified according to Rumelhart, Hinton, and Wiliams (1986)

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) + \alpha \Delta w_{ij}(n-1) \tag{18.29}$$

or according to Sejnowski and Rosenberg (1987)

$$w_{ij}(n+1) = w_{ij}(n) + (1-\alpha)\Delta w_{ij}(n) + \alpha \Delta w_{ij}(n-1) \tag{18.30}$$

where α is the momentum term.

The backpropagation algorithm can be significantly sped up, when, after finding components of the gradient, weights are modified along the gradient direction until a minimum is reached. This process can be carried on without the necessity of a computationally intensive gradient calculation at each step. The new gradient components are calculated once a minimum is obtained in the direction of the previous gradient. This process is only possible for cumulative weight adjustment. One method of finding a minimum along the gradient direction is the *tree step process* of finding error for three points along gradient direction and then, using a parabola approximation, jump directly to the minimum. The fast learning algorithm using the described approach was proposed by Fahlman (1988) and is known as the *quickprop*.

The backpropagation algorithm has many disadvantages, which lead to very slow convergency. One of the most painful is that in the backpropagation algorithm, the learning process almost perishes for neurons responding with the maximally wrong answer. For example, if the value on the neuron output is close to $+1$ and desired output should be close to -1, then the neuron gain $f'(net) \approx 0$ and the error signal cannot backpropagate, and so the learning procedure is not effective. To overcome this difficulty, a modified method for derivative calculation was introduced by Wilamowski and Torvik (1993). The derivative is calculated as the slope of a line connecting the point of the output value with the point of the desired value, as shown in Fig. 18.10.

$$f_{\text{modif}} = \frac{o_{\text{desired}} - o_{\text{actual}}}{net_{\text{desired}} - net_{\text{actual}}} \tag{18.31}$$

Note that for small errors, Eq. (18.31) converges to the derivative of activation function at the point of the output value. With an increase of system dimensionality, the chances for local minima decrease. It is believed that the described phenomenon, rather than a trapping in local minima, is responsible for convergency problems in the error backpropagation algorithm.

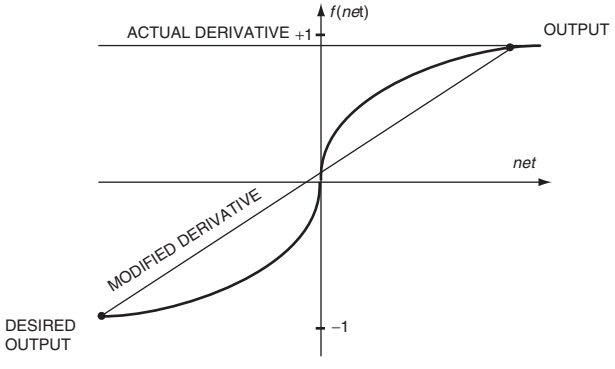


FIGURE 18.10 Illustration of the modified derivative calculation for faster convergency of the error backpropagation algorithm.

© 2006 by Taylor & Francis Group, LLC

18.5 Special Feedforward Networks

The multilayer backpropagation network, as shown in Fig. 18.5, is a commonly used feedforward network. This network consists of neurons with the sigmoid type continuous activation function presented in Fig. 18.4(c) and Fig. 18.4(d). In most cases, only one hidden layer is required, and the number of neurons in the hidden layer are chosen to be proportional to the problem complexity. The number of neurons in the hidden layer is usually found by a trial-and-error process. The training process starts with all weights randomized to small values, and the error backpropagation algorithm is used to find a solution. When the learning process does not converge, the training is repeated with a new set of randomly chosen weights. Nguyen and Widrow (1990) proposed an experimental approach for the two-layer network weight initialization. In the second layer, weights are randomly chosen in the range from -0.5 to $+0.5$. In the first layer, initial weights are calculated from

$$w_{ij} = \frac{\beta z_{ij}}{\|z_j\|}; \qquad w_{(n+1)j} = \text{random}(-\beta, +\beta) \tag{18.32}$$

where z_{ij} is the random number from -0.5 to $+0.5$ and the scaling factor β is given by

$$\beta = 0.7P^{\frac{1}{N}} \tag{18.33}$$

where n is the number of inputs and N is the number of hidden neurons in the first layer. This type of weight initialization usually leads to faster solutions.

For adequate solutions with backpropagation networks, typically many tries are required with different network structures and different initial random weights. It is important that the trained network gains a generalization property. This means that the trained network also should be able to handle correctly patterns that were not used for training. Therefore, in the training procedure, often some data are removed from the training patterns and then these patterns are used for verification. The results with backpropagation networks often depend on luck. This encouraged researchers to develop feedforward networks, which can be more reliable. Some of those networks are described in the following sections.

Functional Link Network

One-layer neural networks are relatively easy to train, but these networks can solve only linearly separated problems. One possible solution for nonlinear problems presented by Nilsson (1965) and elaborated by Pao (1989) using the functional link network is shown in Fig. 18.11. Using nonlinear terms with initially determined functions, the actual number of inputs supplied to the one-layer neural network is increased. In the simplest case, nonlinear elements are higher order terms of input patterns. Note that the functional link network can be treated as a one-layer network, where additional input data are generated off line using nonlinear transformations. The learning procedure for one-layer is easy and fast. Figure 18.12 shows an *X*OR problem solved using functional link networks. Note that when the functional link approach is used, this difficult problem becomes a trivial one. The problem with the functional link network is that proper selection of nonlinear elements is not an easy task. In many practical cases, however, it is not difficult to predict what kind of transformation of input data may linearize the problem, and so the functional link approach can be used.

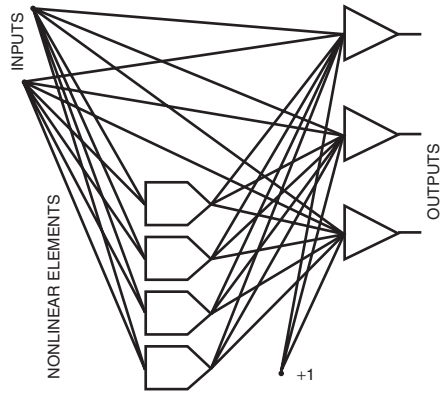


FIGURE 18.11 The functional link network.

© 2006 by Taylor & Francis Group, LLC

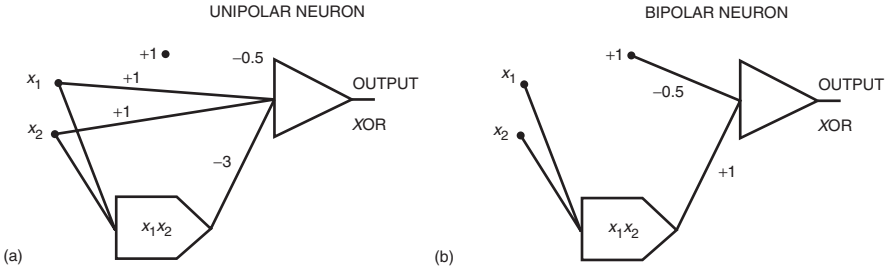


FIGURE 18.12 Functional link networks for solution of the *X*OR problem: (a) using unipolar signals, (b) using bipolar signals.

Feedforward Version of the Counterpropagation Network

The *counterpropagation network* was originally proposed by Hecht-Nilsen (1987). In this section a modified feedforward version as described by Zurada (1992) is discussed. This network, which is shown in Fig. 18.13, requires numbers of hidden neurons equal to the number of input patterns, or more exactly, to the number of input clusters. The first layer is known as the Kohonen layer with unipolar neurons. In this layer only one neuron, the winner, can be active. The second is the Grossberg outstar layer. The Kohonen layer can be trained in the unsupervised mode, but that need not be the case. When binary input patterns are considered, then the input weights must be exactly equal to the input patterns. In this case,

$$net = \boldsymbol{x}^t \boldsymbol{w} = [n - 2HD(\boldsymbol{x}, \boldsymbol{w})] \tag{18.34}$$

where

n = number of inputs
$\boldsymbol{w}$ = weights
$\boldsymbol{x}$ = input vector
$HD(\boldsymbol{w}, \boldsymbol{x})$ = *Hamming distance* between input pattern and weights

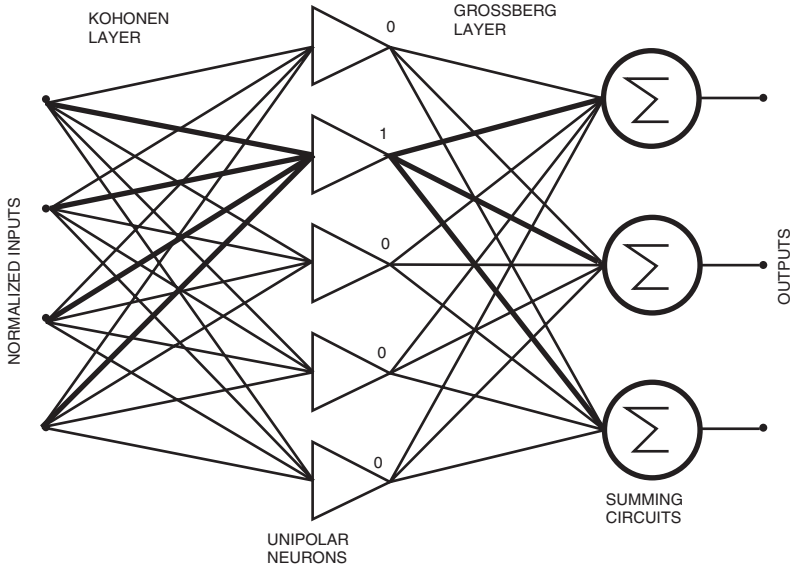


FIGURE 18.13 The counterpropagation network.

© 2006 by Taylor & Francis Group, LLC

For a neuron in the input layer to be reacting just for the stored pattern, the threshold value for this neuron should be

$$w_{(n+1)} = -(n-1) \tag{18.35}$$

If it is required that the neuron must also react for similar patterns, then the threshold should be set to $w_{n+1} = -[n-(1+HD)]$, where HD is the Hamming distance defining the range of similarity. Since for a given input pattern only one neuron in the first layer may have the value of 1 and remaining neurons have 0 values, the weights in the output layer are equal to the required output pattern.

The network, with unipolar activation functions in the first layer, works as a lookup table. When the linear activation function (or no activation function at all) is used in the second layer, then the network also can be considered as an analog memory. For the address applied to the input as a binary vector, the stored set of analog values, as weights in the second layer, can be accurately recovered. The feedforward counterpropagation network may also use analog inputs, but in this case all input data should be normalized.

$$\boldsymbol{w}_i = \hat{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} \tag{18.36}$$

The counterpropagation network is very easy to design. The number of neurons in the hidden layer is equal to the number of patterns (clusters). The weights in the input layer are equal to the input patterns, and the weights in the output layer are equal to the output patterns. This simple network can be used for rapid prototyping. The counterpropagation network usually has more hidden neurons than required. However, such an excessive number of hidden neurons are also used in more sophisticated feedforward networks such as the *probabilistic neural network* (PNN) Specht (1990) or the *general regression neural networks* (GRNN) Specht (1992).

WTA Architecture

The winner takes all network was proposed by Kohonen (1988). This is basically a one-layer network used in the unsupervised training algorithm to extract a statistical property of the input data (Fig. 18.14(a)). At the first step, all input data are normalized so that the length of each input vector is the same and, usually, equal to unity (Eq. (18.36)). The activation functions of neurons are unipolar and continuous. The learning process starts with a weight initialization to small random values. During the learning process the weights are changed only for the neuron with the highest value on the output-the winner.

$$\Delta \boldsymbol{w}_w = c(\boldsymbol{x} - \boldsymbol{w}_w) \tag{18.37}$$

where

$\boldsymbol{w}_w$ = weights of the winning neuron
$\boldsymbol{x}$ = input vector
c = learning constant

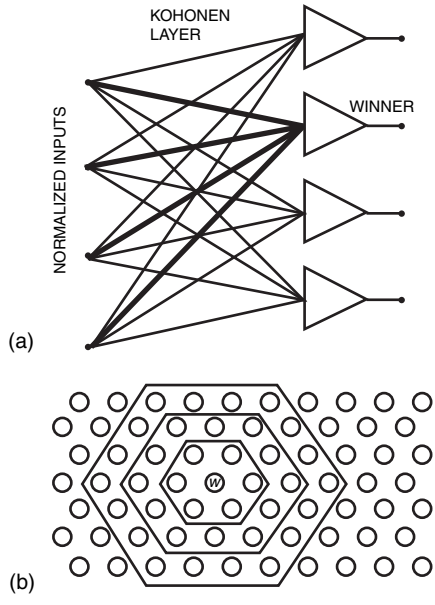


FIGURE 18.14 A winner takes all architecture for cluster extracting in the unsupervised training mode: (a) Network connections, (b) single-layer network arranged into a hexagonal shape.

Usually, this single-layer network is arranged into a two-dimensional layer shape, as shown in Fig. 18.14(b). The hexagonal shape is usually chosen to secure strong interaction between neurons. Also, the algorithm is modified in such a way that not only the winning neuron but also neighboring neurons are allowed for the weight change. At the same time, the learning constant c in Eq. (18.37) decreases with the distance from the winning neuron. After such a unsupervised training procedure, the Kohonen layer is

© 2006 by Taylor & Francis Group, LLC

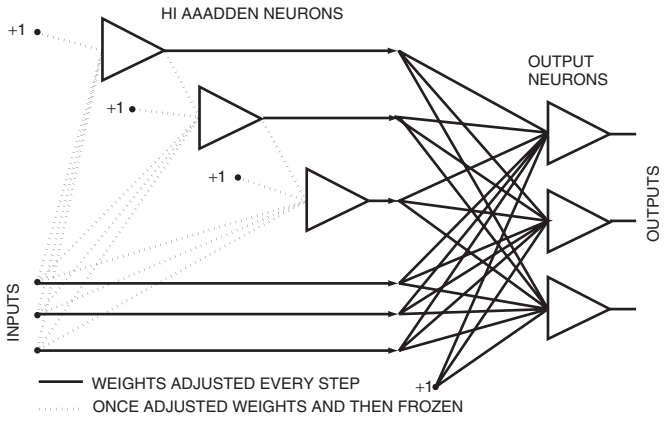


FIGURE 18.15 The cascade correlation architecture.

able to organize data into clusters. Output of the Kohonen layer is then connected to the one- or two-layer feedforward network with the error backpropagation algorithm. This initial data organization in the WTA layer usually leads to rapid training of the following layer or layers.

Cascade Correlation Architecture

The cascade correlation architecture was proposed by Fahlman and Lebiere (1990). The process of network building starts with a one-layer neural network and hidden neurons are added as needed. The network architecture is shown in Fig. 18.15. In each training step, a new hidden neuron is added and its weights are adjusted to maximize the magnitude of the correlation between the new hidden neuron output and the residual error signal on the network output to be eliminated. The correlation parameter S must be maximized.

$$S = \sum_{o=1}^{O} \left| \sum_{p=1}^{P} \left(V_p - \bar{V}\right)\left(E_{po} - \bar{E}_o\right) \right| \tag{18.38}$$

where

O = number of network outputs
P = number of training patterns
V_p = output on the new hidden neuron
E_{po} = error on the network output

$\bar{V}$ and $\bar{E}_o$ are average values of V_p and E_{po}, respectively. By finding the gradient, $\delta S/\delta w_i$, the weight adjustment for the new neuron can be found as

$$\Delta w_i = \sum_{o=1}^{O} \sum_{p=1}^{P} \sigma_o \left(E_{po} - \bar{E}_o\right) f'_p x_{ip} \tag{18.39}$$

where

σ_o = sign of the correlation between the new neuron output value and network output
f'_p = derivative of activation function for pattern p
x_{ip} = input signal

The output neurons are trained using the delta or quickprop algorithms. Each hidden neuron is trained just once and then its weights are frozen. The network learning and building process is completed when satisfactory results are obtained.

© 2006 by Taylor & Francis Group, LLC

Radial Basis Function Networks

The structure of the radial basis network is shown in Fig. 18.16. This type of network usually has only one hidden layer with special neurons. Each of these neurons responds only to the inputs signals close to the stored pattern. The output signal h_i of the ith hidden neuron is computed using formula

$$h_i = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{s}_i\|^2}{2\sigma^2}\right) \tag{18.40}$$

where

$\boldsymbol{x}$ = input vector
$\boldsymbol{s}_i$ = stored pattern representing the center of the i cluster
σ_i = radius of the cluster

Note that the behavior of this "neuron" significantly differs form the biological neuron. In this "neuron," excitation is not a function of the weighted sum of the input signals. Instead, the distance between the input and stored pattern is computed. If this distance is zero then the neuron responds with a maximum output magnitude equal to one. This neuron is capable of recognizing certain patterns and generating output signals that are functions of a similarity. Features of this neuron are much more powerful than a neuron used in the backpropagation networks. As a consequence, a network made of such neurons is also more powerful.

If the input signal is the same as a pattern stored in a neuron, then this neuron responds with 1 and remaining neurons have 0 on the output, as is illustrated in Fig. 18.16. Thus, output signals are exactly equal to the weights coming out from the active neuron. This way, if the number of neurons in the hidden layer is large, then any input/output mapping can be obtained. Unfortunately, it may also happen that for some patterns several neurons in the first layer will respond with a nonzero signal. For a proper approximation, the sum of all signals from the hidden layer should be equal to one. To meet this requirement, output signals are often normalized as shown in Fig. 18.16.

The radial-based networks can be designed or trained. Training is usually carried out in two steps. In the first step, the hidden layer is usually trained in the unsupervised mode by choosing the best patterns for cluster representation. An approach, similar to that used in the WTA architecture can be used. Also in this step, radii σ_i must be found for a proper overlapping of clusters.

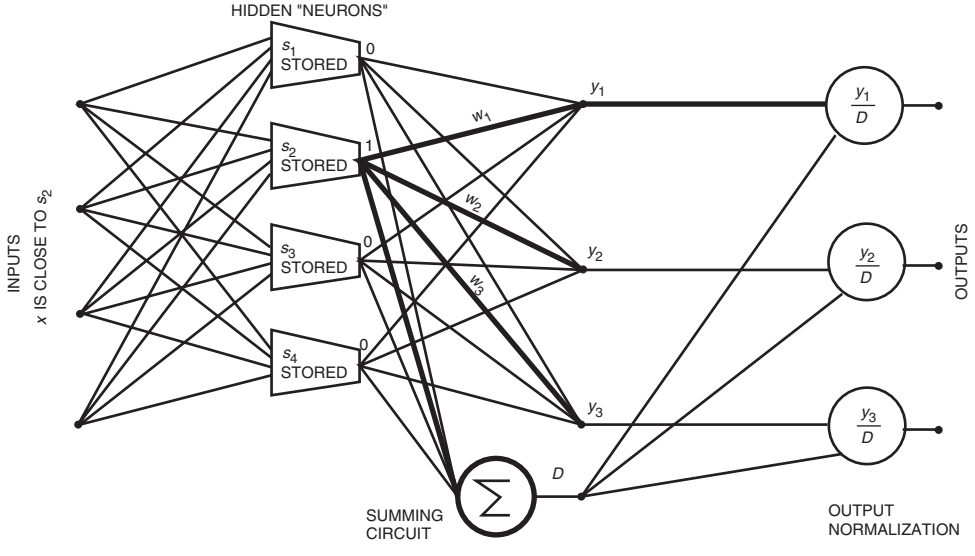


FIGURE 18.16 A typical structure of the radial basis function network.

© 2006 by Taylor & Francis Group, LLC

The second step of training is the error backpropagation algorithm carried on only for the output layer. Since this is a supervised algorithm for one layer only, the training is very rapid, 100–1000 times faster than in the backpropagation multilayer network. This makes the radial basis-function network very attractive. Also, this network can be easily modeled using computers, however, its hardware implementation would be difficult.

18.6 Recurrent Neural Networks

In contrast to feedforward neural networks, with **recurrent networks** neuron outputs can be connected with their inputs. Thus, signals in the network can continuously circulate. Until recently, only a limited number of recurrent neural networks were described.

Hopfield Network

The single-layer recurrent network was analyzed by Hopfield (1982). This network, shown in Fig. 18.17, has unipolar hard threshold neurons with outputs equal to 0 or 1. Weights are given by a symmetrical square matrix $\boldsymbol{W}$ with zero elements ($w_{ij} = 0$ for $i = j$) on the main diagonal. The stability of the system is usually analyzed by means of the *energy function*

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} v_i v_j \tag{18.41}$$

It has been proved that during signal circulation the energy E of the network decreases and the system converges to the stable points. This is especially true when the values of system outputs are updated in the asynchronous mode. This means that at a given cycle, only one random output can be changed to the required values. Hopfield also proved that those stable points which the system converges can be programmed by adjusting the weights using a modified Hebbian rule,

$$\Delta w_{ij} = \Delta w_{ji} = (2v_i - 1)(2v_j - 1)c \tag{18.42}$$

Such memory has limited storage capacity. Based on experiments, Hopfield estimated that the maximum number of stored patterns is $0.15N$, where N is the number of neurons.

Later the concept of energy function was extended by Hopfield (1984) to one-layer recurrent networks having neurons with continuous activation functions. These types of networks were used to solve many optimization and linear programming problems.

Autoassociative Memory

Hopfield (1984) extended the concept of his network to autoassociative memories. In the same network structure as shown in Fig. 18.17, the bipolar hard-threshold neurons were used with outputs equal to -1 or $+1$. In this network, pattern $\boldsymbol{s}_m$ are stored into the weight matrix $\boldsymbol{W}$ using the autocorrelation algorithm

$$\boldsymbol{W} = \sum_{m=1}^{M} \mathbf{s}_m \mathbf{s}_m^T - M\boldsymbol{I} \tag{18.43}$$

where M is the number of stored pattern and $\boldsymbol{I}$ is the unity matrix. Note that $\boldsymbol{W}$ is the square symmetrical matrix with elements on the main diagonal equal to zero (w_{ji} for $i = j$). Using a modified formula (18.42), new patterns can be added or subtracted from memory. When such memory is exposed to a binary bipolar pattern by enforcing the initial network states, after signal circulation the network will converge to the

© 2006 by Taylor & Francis Group, LLC

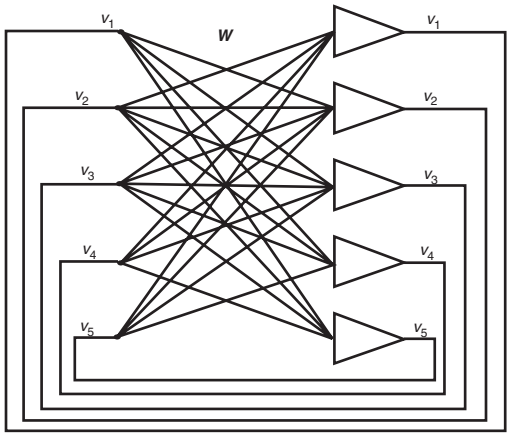


FIGURE 18.17 A Hopfield network or autoassociative memory.

closest (most similar) stored pattern or to its complement. This stable point will be at the closest minimum of the energy

$$E(\boldsymbol{v}) = -\frac{1}{2}\boldsymbol{v}^T\mathbf{W}\boldsymbol{v} \tag{18.44}$$

Like the Hopfield network, the autoassociative memory has limited storage capacity, which is estimated to be about $M_{\max} = 0.15N$. When the number of stored patterns is large and close to the memory capacity, the network has a tendency to converge to spurious states, which were not stored. These spurious states are additional minima of the energy function.

Bidirectional Associative Memories (BAM)

The concept of the autoassociative memory was extended to bidirectional associative memories (BAM) by Kosko (1987, 1988). This memory, shown in Fig. 18.18, is able to associate pairs of the patterns $\boldsymbol{a}$ and $\boldsymbol{b}$. This is the two-layer network with the output of the second layer connected directly to the input of the first layer. The weight matrix of the second layer is $\boldsymbol{W}^T$ and $\boldsymbol{W}$ for the first layer. The rectangular weight

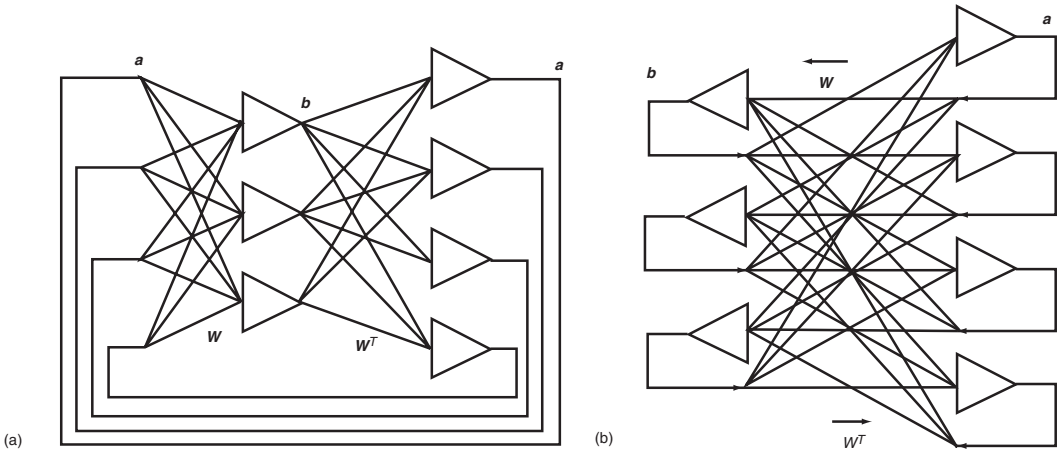


FIGURE 18.18 An example of the bi-directional autoassociative memory: (a) drawn as a two-layer network with circulating signals, (b) drawn as two-layer network with bi-directional signal flow.

© 2006 by Taylor & Francis Group, LLC

matrix $\boldsymbol{W}$ is obtained as a sum of the cross-correlation matrixes

$$\boldsymbol{W} = \sum_{m=1}^{M} \boldsymbol{a}_m \boldsymbol{b}_m \tag{18.45}$$

where M is the number of stored pairs, and $\boldsymbol{a}_m$ and $\boldsymbol{b}_m$ are the stored vector pairs. If the nodes $\boldsymbol{a}$ or $\boldsymbol{b}$ are initialized with a vector similar to the stored one, then after signal circulations, both stored patterns $\boldsymbol{a}_m$ and $\boldsymbol{b}_m$ should be recovered. The BAM has limited memory capacity and memory corruption problems similar to the autoassociative memory. The BAM concept can be extended for association of three or more vectors.

18.7 Fuzzy Systems

The main applications of neural networks are related to the nonlinear mapping of n-dimensional input variables into m-dimensional output variables. Such a function is often required in control systems, where for specific measured variables certain control variables must be generated. Another approach for nonlinear mapping of one set of variables into another set of variables is the *fuzzy controller*. The principle of operation of the fuzzy controller significantly differs from neural networks. The block diagram of a fuzzy controller is shown in Fig. 18.19. In the first step, analog inputs are converted into a set of fuzzy variables. In this step, for each analog input, 3–9 fuzzy variables typically are generated. Each fuzzy variable has an analog value between zero and one. In the next step, a fuzzy logic is applied to the input fuzzy variables and a resulting set of output variables is generated. In the last step, known as *defuzzification*, from a set of output fuzzy variables, one or more output analog variables are generated, which are used as control variables.

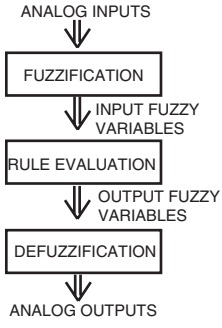


FIGURE 18.19 The block diagram of the fuzzy controller.

Fuzzification

The purpose of fuzzification is to convert an analog variable input into a set of fuzzy variables. For higher accuracy, more fuzzy variables will be chosen. To illustrate the fuzzification process, consider that the input variable is the temperature and is coded into five fuzzy variables: cold, cool, normal, warm, and hot. Each fuzzy variable should obtain a value between zero and one, which describes a *degree of association* of the analog input (temperature) within the given fuzzy variable. Sometimes, instead of the term degree of association, the term *degree of membership* is used. The process of fuzzification is illustrated in Fig. 18.20. Using Fig. 18.20 we can find the degree of association of each fuzzy variable with the given temperature. For example, for a temperature of 57°F, the following set of fuzzy variables is obtained: (0, 0.5, 0.2, 0, 0), and for $T = 80$°F it is (0, 0, 0.25, 0.7, 0). Usually only one or two fuzzy variables have a value other than

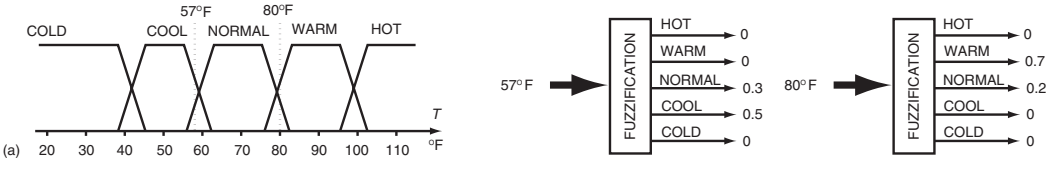


FIGURE 18.20 Fuzzification process: (a) typical membership functions for the fuzzification and the defuzzification processes, (b) example of converting a temperature into fuzzy variables.

© 2006 by Taylor & Francis Group, LLC

zero. In the example, trapezoidal functions are used for calculation of the degree of association. Various different functions such as triangular or Gaussian, can also be used, as long as the computed value is in the range from zero to one. Each membership function is described by only three or four parameters, which have to be stored in memory.

For proper design of the fuzzification stage, certain practical rules should be used:

- Each point of the input analog variable should belong to at least one and no more than two membership functions.
- For overlapping functions, the sum of two membership functions must not be larger than one. This also means that overlaps must not cross the points of maximum values (ones).
- For higher accuracy, more membership functions should be used. However, very dense functions lead to frequent system reaction and sometimes to system instability.

Rule Evaluation

Contrary to Boolean logic where variables can have only binary states, in fuzzy logic all variables may have any values between zero and one. The fuzzy logic consists of the same basic $\wedge$ - AND, $\vee$-OR, and NOT operators

$$A \wedge B \wedge C \quad \Longrightarrow \min\{A, B, C\}\text{—smallest value of } A \text{ or } B \text{ or } C$$
$$A \vee B \vee C \quad \Longrightarrow \max\{A, B, C\}\text{—largest value of } A \text{ or } B \text{ or } C$$
$$\bar{A} \quad \Longrightarrow 1\ 1 - A \qquad \text{—one minus value of } A$$

For example $0.1 \wedge 0.7 \wedge 0.3 = 0.1, 0.1 \vee 0.7 \vee 0.3 = 0.7$, and $\bar{0.3} = 0.7$. These rules are also known as *Zadeh* AND, OR, and NOT operators (Zadeh, 1965). Note that these rules are true also for classical binary logic.

Fuzzy rules are specified in the *fuzzy table* as it is shown for a given system. Consider a simple system with two analog input variables x and y, and one output variable z. The goal is to design a fuzzy system generating z as $f(x, y)$. After fuzzification, the analog variable x is represented by five fuzzy variables: x_1, x_2, x_3, x_4, x_5 and an analog variable y is represented by three fuzzy variables: y_1, y_2, y_3. Assume that an analog output variable is represented by four fuzzy variables: z_1, z_2, z_3, z_4. The key issue of the design process is to set proper output fuzzy variables z_k for all combinations of input fuzzy variables, as is shown in the table in Fig. 18.21. The designer has to specify many rules such as if inputs are represented by fuzzy variables x_i and y_j, then the output should be represented by fuzzy variable z_k. Once the fuzzy table is specified, the fuzzy logic computation proceeds in two steps. First each field of the fuzzy table is filled with intermediate fuzzy variables t_{ij}, obtained from AND operator $t_{ij} = \min\{x_i, y_j\}$, as shown in Fig. 18.21(b). This step is independent of the required rules for a given system. In the second step, the OR (max) operator is used to compute each output fuzzy variable z_k. In the given example in Fig. 18.21, $z_1 = \max\{t_{11}, t_{12}, t_{21}, t_{41}, t_{51}\}$, $z_2 = \max\{t_{13}, t_{31}, t_{42}, t_{52}\}$, $z_3 = \max\{t_{22}, t_{23}, t_{43}\}$, $z_4 = \max\{t_{32}, t_{34}, t_{53}\}$. Note that the formulas depend on the specifications given in the fuzzy table shown in Fig. 18.21(a).

(a)

	y_1	y_2	y_3
x_1	z_1	z_1	z_2
x_2	z_1	z_3	z_3
x_3	z_2	z_4	z_4
x_4	z_1	z_2	z_3
x_5	z_1	z_2	z_4

(b)

	y_1	y_2	y_3
x_1	t_{11}	t_{12}	t_{13}
x_2	t_{21}	t_{22}	t_{23}
x_3	t_{31}	t_{32}	t_{33}
x_4	t_{41}	t_{42}	t_{43}
x_5	t_{51}	t_{52}	t_{53}

FIGURE 18.21 Fuzzy tables: (a) table with fuzzy rules, (b) table with the intermediate variables t_{ij}.

Defuzzification

As a result of fuzzy rule evaluation, each analog output variable is represented by several fuzzy variables. The purpose of defuzzification is to obtain analog outputs. This can be done by using a membership function similar to that shown in Fig. 18.20. In the first step, fuzzy variables obtained from rule evaluations are used

© 2006 by Taylor & Francis Group, LLC

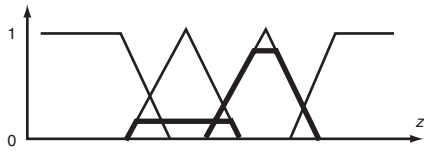


FIGURE 18.22 Illustration of the defuzzification process.

to modify the membership function employing the formula

$$\mu_k^*(z) = \min\{\mu_k(z), z_k\} \tag{18.46}$$

For example, if the output fuzzy variables are: 0, 0.2, 0.7, 0.0, then the modified membership functions have shapes shown by the thick line in Fig. 18.22. The analog value of the z variable is found as a *center of gravity* of modified membership functions $\mu_k^*(z)$,

$$z_{\text{analog}} = \frac{\left(\sum_{k=1}^{n} \int_{-\infty}^{+\infty} \mu_k{}^*(z) z dz\right)}{\left(\sum_{k=1}^{n} \int_{-\infty}^{+\infty} \mu_k{}^*(z) dz\right)} \tag{18.47}$$

In the case where shapes of the output membership functions $\mu_k(z)$ are the same, the equation can be simplified to

$$z_{\text{analog}} = \frac{\left(\sum_{k=1}^{n} z_k zc_k\right)}{\left(\sum_{k=1}^{n} z_k\right)} \tag{18.48}$$

where

n = number of membership functions of z_{analog} output variable
z_k = fuzzy output variables obtained from rule evaluation
zc_k = analog values corresponding to the center of kth membership function.

Equation (18.47) is usually too complicated to be used in a simple microcontroller based system; therefore, in practical cases, Eq. (18.48) is used more frequently.

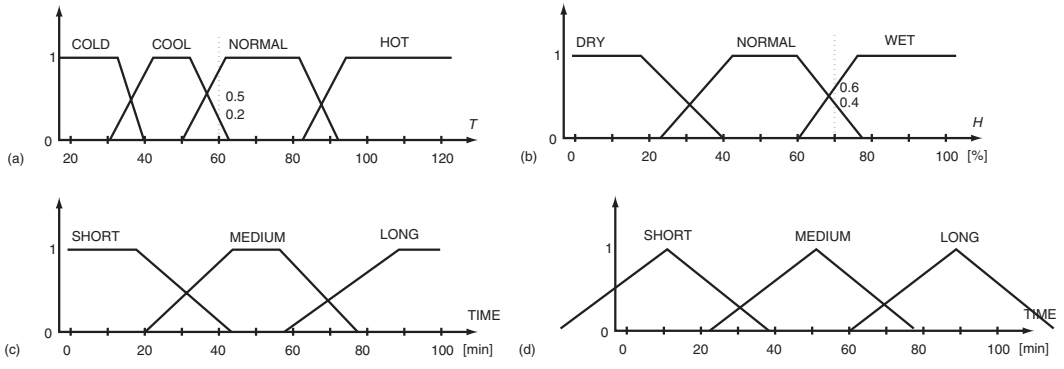


FIGURE 18.23 Membership functions for the presented example: (a) and (b) are membership functions for input variables, (c) and (d) are two possible membership functions for the output variable.

© 2006 by Taylor & Francis Group, LLC

18.8 Design Example

Consider the design of a simple fuzzy controller for a sprinkler system. The sprinkling time is a function of humidity and temperature. Four membership functions are used for the temperature, three for humidity, and three for the sprinkle time, as shown in Fig. 18.23. Using intuition, the fuzzy table can be developed, as shown in Fig. 18.24(a).

Assume a temperature of 60°F and 70% humidity. Using the membership functions for temperature and humidity the following fuzzy variables can be obtained for the temperature: (0, 0.2, 0.5, 0), and for the humidity: (0, 0.4, 0.6). Using the min operator, the fuzzy table can be now filled with temporary fuzzy variables, as shown in Fig. 18.24(b). Note that only four fields have nonzero values. Using fuzzy rules, as shown in Fig. 18.24(a), the max operator can be applied in order to obtain fuzzy output variables: short $\rightarrow o_1 = \max\{0, 0, 0.2, 0.5, 0\} = 0.5$, medium $\rightarrow o_2 = \max\{0, 0, 0.2, 0.4, 0\} = 0.4$, long $\rightarrow o_3 = \max\{0, 0\} = 0$. Using Eq. (18.47) and Fig. 18.23(c) a sprinkle time of 28 min is determined. When the simplified approach is used with Eq. (18.46) and Fig. 18.23(d), then sprinkle time is 27 min.

(a)

	DRY	NORMAL	WET
COLD	M	S	S
COOL	M	M	S
WARM	L	M	S
HOT	L	M	S

(b)

	DRY	NORMAL	WET
	0	0.4	0.6
COLD 0	0	0	0
COOL 0.2	0	0.2	0.2
WARM 0.5	0	0.4	0.5
HOT 0	0	0	0

FIGURE 18.24 Fuzzy tables: (a) fuzzy rules for the design example, (b) fuzzy temporary variables for the design example.

18.9 Genetic Algorithms

The success of the artificial neural networks encouraged researchers to search for other patterns in nature to follow. The power of genetics through evolution was able to create such sophisticated machines as the human being. Genetic algorithms follow the evolution process in nature to find better solutions to some complicated problems. The foundations of genetic algorithms are given by Holland (1975) and Goldberg (1989). After initialization, the steps *selection, reproduction with a crossover*, and *mutation* are repeated for each generation. During this procedure, certain strings of symbols, known as chromosomes, evaluate toward a better solution. The genetic algorithm method begins with coding and an initialization. All significant steps of the genetic algorithm will be explained using a simple example of finding a maximum of the function $(\sin^2 x - 0.5x)^2$ (Fig. 18.25) with the range of x from 0 to 1.6. Note that in this range, the function has a global maximum at $x = 1.309$, and a local maximum at $x = 0.262$.

Coding and Initialization

At first, the variable x has to be represented as a string of symbols. With longer strings, the process usually converges faster, so the fewer the symbols for one string field that are used, the better. Although this string may be the sequence of any symbols, the binary symbols 0 and 1 are usually used. In our example, six bit binary numbers are used for coding, having a decimal value of $40x$. The process starts with a random generation of the initial population given in Table 18.1.

Selection and Reproduction

Selection of the best members of the population is an important step in the genetic algorithm. Many different approaches can be used to rank individuals. In this example the ranking function is given. Highest rank has member number 6, and lowest rank has member number 3. Members with higher rank should have higher chances to reproduce. The probability of reproduction for each member can be obtained as a

© 2006 by Taylor & Francis Group, LLC

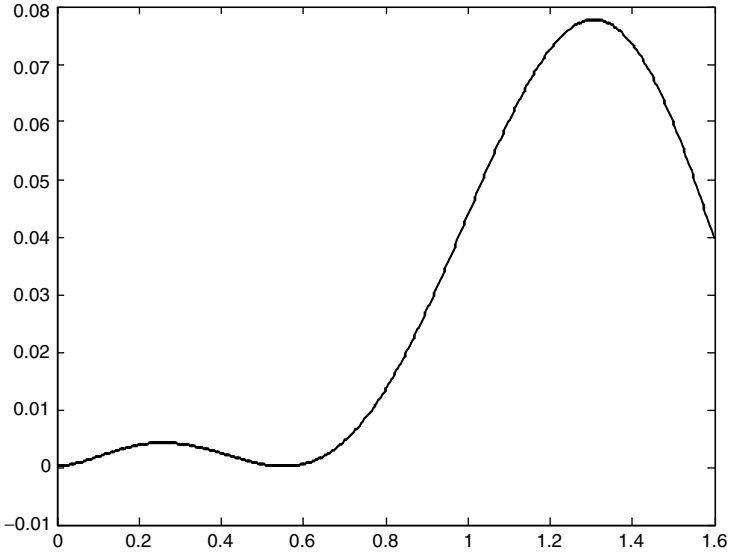


FIGURE 18.25 Function ($\sin^2 x - 0.5x^2$) for which the minimum must be sought.

TABLE 18.1 Initial Population

String number	String	Decimal value	Variable value	Function value	Fraction of total
1	101101	45	1.125	0.0633	0.2465
2	101000	40	1.000	0.0433	0.1686
3	010100	20	0.500	0.0004	0.0016
4	100101	37	0.925	0.0307	0.1197
5	001010	10	0.250	0.0041	0.0158
6	110001	49	1.225	0.0743	0.2895
7	100111	39	0.975	0.0390	0.1521
8	000100	40	0.100	0.0016	0.0062
Total				0.2568	1.0000

fraction of the sum of all objective function values. This fraction is shown in the last column of Table 18.1. Note that to use this approach, our objective function should always be positive. If it is not, the proper normalization should be introduced at first.

Reproduction

The numbers in the last column of Table 18.1 show the probabilities of reproduction. Therefore, most likely members number 3 and 8 will not be reproduced, and members 1 and 6 may have two or more copies. Using a random reproduction process, the following population, arranged in pairs, could be generated

$$101101 \rightarrow 45 \quad 110001 \rightarrow 49 \quad 100101 \rightarrow 37 \quad 110001 \rightarrow 49$$
$$100111 \rightarrow 39 \quad 101101 \rightarrow 45 \quad 110001 \rightarrow 49 \quad 101000 \rightarrow 40$$

If the size of the population from one generation to another is the same, two parents should generate two children. By combining two strings, two other strings should be generated. The simplest way to do this

© 2006 by Taylor & Francis Group, LLC

is to split in half each of the parent strings and exchange substrings between parents. For example, from parent strings 010100 and 100111, the following child strings will be generated 010111 and 100100. This process is known as the *crossover*. The resultant children are

$$\begin{array}{llll} 101111 \rightarrow 47 & 110101 \rightarrow 53 & 100001 \rightarrow 33 & 110000 \rightarrow 48 \\ 100101 \rightarrow 37 & 101001 \rightarrow 41 & 110101 \rightarrow 53 & 101001 \rightarrow 41 \end{array}$$

In general, the string need not be split in half. It is usually enough if only selected bits are exchanged between parents. It is only important that bit positions are not changed.

Mutation

In the evolutionary process, reproduction is enhanced with mutation. In addition to the properties inherited from parents, offspring acquire some new random properties. This process is known as mutation. In most cases mutation generates low-ranked children, which are eliminated in the reproduction process. Sometimes, however, the mutation may introduce a better individual with a new property. This prevents the process of reproduction from degeneration. In genetic algorithms, mutation usually plays a secondary role. For very high-levels of mutation, the process is similar to random pattern generation, and such a searching algorithm is very inefficient. The mutation rate is usually assumed to be at a level well below 1%. In this example, mutation is equivalent to the random bit change of a given pattern. In this simple case, with short strings and a small population, and with a typical mutation rate of 0.1%, the patterns remain practically unchanged by the mutation process. The second generation for this example is shown in Table 18.2.

Note that two identical highest ranking members of the second generation are very close to the solution $x = 1.309$. The randomly chosen parents for the third generation are

$$\begin{array}{llll} 010111 \rightarrow 47 & 110101 \rightarrow 53 & 110000 \rightarrow 48 & 101001 \rightarrow 41 \\ 110101 \rightarrow 53 & 110000 \rightarrow 48 & 101001 \rightarrow 41 & 110101 \rightarrow 53 \end{array}$$

which produces the following children

$$\begin{array}{llll} 010101 \rightarrow 21 & 110000 \rightarrow 48 & 110001 \rightarrow 49 & 101101 \rightarrow 45 \\ 110111 \rightarrow 55 & 110101 \rightarrow 53 & 101000 \rightarrow 40 & 110001 \rightarrow 49 \end{array}$$

The best result in the third population is the same as in the second one. By careful inspection of all strings from the second or third generation, it may be concluded that using crossover, where strings are always split in half, the best solution 110100 $\rightarrow$52 will never be reached, regardless of how many generations are created. This is because none of the population in the second generation has a substring ending with 100. For such crossover, a better result can be only obtained due to the mutation process, which may require

TABLE 18.2 Population of Second Generation

String number	String	Decimal value	Variable value	Function value	Fraction of total
1	010111	47	1.175	0.0696	0.1587
2	100100	37	0.925	0.0307	0.0701
3	110101	53	1.325	0.0774	0.1766
4	010001	41	1.025	0.0475	0.1084
5	100001	33	0.825	0.0161	0.0368
6	110101	53	1.325	0.0774	0.1766
7	110000	48	1.200	0.0722	0.1646
8	101001	41	1.025	0.0475	0.1084
Total				0.4387	1.0000

© 2006 by Taylor & Francis Group, LLC

many generations. Better results in the future generation also can be obtained when strings are split in random places. Another possible solution is that only randomly chosen bits are exchanged between parents.

The genetic algorithm almost always leads to a good solution, but sometimes many generations are required. This solution is usually close to global maximum, but not the best. In the case of a smooth function the gradinet methods are converging much faster and to a better solution. GA are much slower, but more robust.

Defining Terms

Backpropagation: Training technique for multilayer neural networks.
Bipolar neuron: Neuron with output signal between -1 and $+1$.
Feedforward network: Network without feedback.
Perceptron: Network with hard threshold neurons.
Recurrent network: Network with feedback.
Supervised learning: Learning procedure when desired outputs are known.
Unipolar neuron: Neuron with output signal between 0 and $+1$.
Unsupervised learning: Learning procedure when desired outputs are unknown.

References

Fahlman, S.E. 1988. Faster-learning variations on backpropagation: an empirical study. *Proceedings of the Connectionist Models Summer School*, eds. Touretzky, D. Hinton, G., and Sejnowski, T. Morgan Kaufmann, San Mateo, CA.

Fahlman, S.E. and Lebiere, C. 1990. The cascade correlation learning architecture. *Adv. Ner. Inf. Proc. Syst.*, 2, D.S. Touretzky, ed. pp. 524–532. Morgan, Kaufmann, Los Altos, CA.

Goldberg, D.E. 1989. *Genetic Algorithm in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Grossberg, S. 1969. Embedding fields: a theory of learning with physiological implications. *Journal of Mathematical Psychology* 6:209–239.

Hebb, D.O. 1949. *The Organization of Behivior, a Neuropsychological Theory*. John Wiley, New York.

Hecht-Nielsen, R. 1987. Counterpropagation networks. *Appl. Opt.* 26(23):4979–4984.

Hecht-Nielsen, R. 1988. Applications of counterpropagation networks. *Neural Networks* 1:131–139.

Holland, J.H. 1975. *Adaptation in Natural and Artificial Systems*. University. of Michigan Press, Ann Arbor, MI.

Hopfield, J.J. 1982. Neural networks and physical systems with emergent collective computation abilities. *Proceedings of the National Academy of Science* 79:2554–2558.

Hopfield, J.J. 1984. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science* 81:3088–3092.

Kohonen, T. 1988. The neural phonetic typerater. *IEEE Computer* 27(3):11–22.

Kohonen, T. 1990. The self-organized map. *Proc. IEEE* 78(9):1464–1480.

Kosko, B. 1987. Adaptive bidirectional associative memories. *App. Opt.* 26:4947–4959.

Kosko, B. 1988. Bidirectional associative memories. *IEEE Trans. Sys. Man, Cyb.* 18:49–60.

McCulloch, W.S. and Pitts, W.H. 1943. A logical calculus of the ideas imminent in nervous activity. *Bull. Math. Biophy.* 5:115–133.

Minsky, M. and Papert, S. 1969. *Perceptrons*. MIT Press, Cambridge, MA.

Nilsson, N.J. 1965. *Learning Machines: Foundations of Trainable Pattern Classifiers*. McGraw-Hill, New York.

Nguyen, D. and Widrow, B. 1990. Improving the learning speed of 2-layer neural networks, by choosing initial values of the adaptive weights. *Proceedings of the International Joint Conference on Neural Networks* (San Diego), CA, June.

Pao, Y.H. 1989. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, Reading, MA.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psych. Rev.* 65:386–408.

© 2006 by Taylor & Francis Group, LLC

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. Learning internal representation by error propagation. *Parallel Distributed Processing*. Vol. 1, pp. 318–362. MIT Press, Cambrige, MA.

Sejnowski, T.J. and Rosenberg, C.R. 1987. Parallel networks that learn to pronounce English text. *Complex Systems* 1:145–168.

Specht, D.F. 1990. Probalistic neural networks. *Neural Networks* 3:109–118.

Specht, D.F. 1992. General regression neural network. *IEEE Trans. Neural Networks* 2:568–576.

Wasserman, P.D. 1989. *Neural Computing Theory and Practice*. Van Nostrand Reinhold, New York.

Werbos, P. 1974. Beyond regression: new tools for prediction and analysis in behavioral sciences. Ph.D. diss., Harvard Universtiy.

Widrow, B. and Hoff, M.E., 1960. Adaptive switching circuits. 1960 IRE Western Electric Show and Convention Record, Part 4 (Aug. 23):96–104.

Widrow, B. 1962. Generalization and information storage in networks of adaline Neurons. In *Self-organizing Systems,* Jovitz, M.C., Jacobi, G.T. and Goldstein, G. eds. pp. 435–461. Sparten Books, Washington, D.C.

Wilamowski, M. and Torvik, L. 1993. Modification of gradient computation in the backpropagation algorithm. *ANNIE'93—Artificial Neural Networks in Engineering*. November 14–17, 1993, St. Louis, Missou.; also in Dagli, C.H. ed. 1993. *Intelligent Engineering Systems Through Artificial Neural Networks* Vol. 3, pp. 175–180. ASME Press, New York.

Zadeh, L.A. 1965. Fuzzy sets. *Information and Control* 8:338–353.

Zurada, J. M. 1992. *Introduction to Artificial Neural Systems*. West Publishing Company, St. Paul. MN.

© 2006 by Taylor & Francis Group, LLC

19

Machine Vision

19.1 Introduction **19**-1
Relationship to Other Fields • Fundamentals of Vision

19.2 Image Formation **19**-3
Imaging Geometry • Image Intensity • Sampling and Quantization • Color Vision • Range Imaging • Structured Lighting • Active Vision

19.3 Segmentation **19**-7
Edge-Based Segmentation • Region-Based Segmentation • Other Segmentation Methods

19.4 Feature Extraction and Matching **19**-12
Representation and Description • Feature Extraction • Matching

19.5 Three-Dimensional Object Recognition **19**-16
Recognition System Components • Object Representation Schemes

19.6 Dynamic Vision **19**-19
Change Detection • Optical Flow • Segmentation Using a Moving Camera

19.7 Applications **19**-23
Optical Character Recognition (OCR) and Document Image Analysis • Medical Image Analysis • Photogrammetry and Aerial Image Analysis • Industrial Inspection and Robotics • Autonomous Navigation • Visual Information Management Systems

David A. Kosiba

Rangachar Kasturi

19.1 Introduction

Machine vision, also known as computer vision, is the scientific discipline whereby explicit, meaningful descriptions of physical objects from the world around us are constructed from their images. Machine vision produces measurements or abstractions from geometrical properties and comprises techniques for estimating features in images, relating feature measurements to the geometry of objects in space, and interpreting this geometric information. This overall task is generally termed *image understanding*.

The goal of a machine vision system is to create a model of the real world from images or sequences of images. Since images are two-dimensional projections of the three-dimensional world, the information is not directly available and must be recovered. This recovery requires the inversion of a many-to-one mapping. To reclaim this information, however, knowledge about the objects in the scene and projection geometry is required.

At every stage in a machine vision system decisions must be made requiring knowledge of the application or goal. Emphasis in machine vision systems is on maximizing the automatic operation at each stage, and these systems should use knowledge to accomplish this. The knowledge used by the system includes models

© 2006 by Taylor & Francis Group, LLC

of features, image formation, objects, and relationships among objects. Without explicit use of knowledge, machine vision systems can be designed to work only in a very constrained environment for limited applications. To provide more flexibility and robustness, knowledge is represented explicitly and is directly used by the system. Knowledge is also used by the designers of machine vision systems in many implicit as well as explicit forms. In fact, the efficacy and efficiency of a system is usually governed by the quality of the knowledge used by the system. Difficult problems are often solvable only by identifying the proper source of knowledge and appropriate mechanisms to use it in the system.

Relationship to Other Fields

Machine vision is closely related to many other disciplines and incorporates various techniques adopted from many well-established fields, such as physics, mathematics, and psychology. Techniques developed from many areas are used for recovering information from images. In this section, we briefly describe some very closely related fields.

Image processing techniques generally transform images into other images; the task of information recovery is left to a human user. This field includes topics such as image enhancement, image compression, and correcting blurred or out-of-focus images (Gonzalez and Woods, 1982). On the other hand, machine vision algorithms take images as inputs but produce other types of outputs, such as representations for the object contours in an image or the motion of objects in a series of images. Thus, emphasis in machine vision is on recovering information automatically, with minimal interaction with a human. Image processing algorithms are useful in the early stages of a machine vision system. They are usually used to enhance particular information and suppress noise.

Computer graphics generates images from geometric primitives such as lines, circles, and free-form surfaces. Computer graphics techniques play a significant role in visualization and virtual reality (Foley et al., 1990). Machine vision is the inverse problem: estimating the geometric primitives and other features from the image. Thus, computer graphics is the synthesis of images, and machine vision is the analysis of images. However, these two fields are growing closer. Machine vision is using curve and surface representations and several other techniques from computer graphics, and computer graphics is using many techniques from machine vision to enter models into the computer for creating realistic images. Visualization and virtual reality are bringing these two fields closer.

Pattern recognition classifies numerical and symbolic data. Many statistical and syntactical techniques have been developed for classification of patterns. Techniques from pattern recognition play an important role in machine vision for recognizing objects. In fact, many vision applications rely heavily on pattern recognition. Object recognition in machine vision usually requires many other techniques. For a complete discussion of pattern recognition techniques, see Duda and Hart (1973).

Artificial intelligence is concerned with designing systems that are intelligent and with studying computational aspects of intelligence (Winston, 1992; Tanimoto, 1995). Artificial intelligence is used to analyze scenes by computing a symbolic representation of the scene contents after the images have been processed to obtain features. Artificial intelligence may be viewed as having three stages: perception, cognition, and action. Perception translates signals from the world into symbols, cognition manipulates the symbols, and action translates the symbols into signals that effect changes in the world. Many techniques from artificial intelligence play important roles in all aspects of machine vision. In fact, machine vision is often considered a subfield of artificial intelligence. Directly related to artificial intelligence is the study of *neural networks*. The design and analysis of neural networks has become a very active field in the last decade (Kosko, 1992). Neural networks are being increasingly applied to solve some machine vision problems.

Psychophysics, along with cognitive science, has studied human vision for a long time (Marr, 1982). Some techniques in machine vision are related to what is known about human vision. Many researchers in computer vision are more interested in preparing computational models of human vision than in designing machine vision systems. Therefore, some of the techniques used in machine vision have a strong similarity to those in psychophysics.

© 2006 by Taylor & Francis Group, LLC

Fundamentals of Vision

Like other developing disciplines, machine vision is based on certain fundamental principles and techniques. To design and develop successful machine vision systems, one must have a full understanding of all aspects of the system, from initial image formation to final scene interpretation. The following is a list of the primary topics in machine vision:

- Image formation
- Segmentation
- Feature extraction and matching
- Three-dimensional object recognition
- Dynamic vision

In each of these processes, numerous factors influence the choice of algorithms and techniques to be used in developing a specific system. The system designer should be knowledgeable about the design issues of the system and the tradeoffs between the various algorithms and techniques available. In the following, we very briefly introduce these topics. Throughout the following sections, we make no attempt to cite all original work as the list would be nearly endless. The references, however, do give credit appropriately, and we refer you to the Further Information section for a complete discussion on these and many other topics.

19.2 Image Formation

An image is formed when a sensor records received radiation as a two-dimensional function. The brightness or intensity values in an image may represent different physical entities. For example, in a typical image obtained by a video camera, the intensity values represent the reflectance of light from various object surfaces in the scene; in a thermal image, they represent the temperature of corresponding regions in the scene; and in range imaging, they represent the distance from the camera to various points in the scene. Multiple images of the same scene are often captured using different types of sensors to facilitate more robust and reliable interpretation of the scene. Selecting an appropriate image formation system plays a key role in the design of practical machine vision systems. The following sections describe the principles of image formation.

Imaging Geometry

A simple camera-centered imaging model is shown in Fig. 19.1. The coordinate system is chosen such that the xy plane is parallel with the image plane and the z axis passes through the lens center at a distance f, the focal length of the camera, from the image plane. The image of a scene point (x, y, z) forms a point (x', y') on the image plane where

$$x' = x\frac{f}{z} \quad \text{and} \quad y' = y\frac{f}{z} \tag{19.1}$$

These are the *perspective projection* equations of the imaging system.

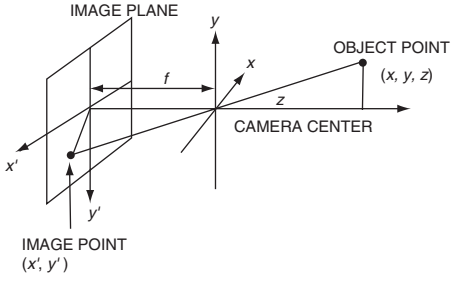


FIGURE 19.1 The camera-centered coordinate system imaging model.

© 2006 by Taylor & Francis Group, LLC

In a typical imaging situation, the camera may have several degrees of freedom, such as translation, pan, and tilt. Also, more than one camera may be imaging the same scene from different points. In this case, it is convenient to adopt a world coordinate system in reference to which the scene coordinates and camera coordinates are defined. In this situation, however, the imaging equations become more cumbersome and we refer you to the references at the end of this chapter for a more complete discussion of imaging geometry including camera calibration.

Image Intensity

Although imaging geometry uniquely determines the relationship between scene coordinates and image coordinates, the brightness or intensity at each point is determined not only by the imaging geometry but also by several other factors, including scene illumination, reflectance properties and surface orientations of objects in the scene, and radiometric properties of the imaging sensor.

The reflectance properties of a surface are characterized by its *bidirectional reflectance distribution function* (BRDF). The BRDF is the ratio of the scene radiance in the direction of the observer to the irradiance due to a light source from a given direction. It captures how bright a surface will appear when viewed from a given direction and illuminated by another. For example, for a flat Lambertian surface illuminated by a distant point light source, the BRDF is constant; hence, the surface appears equally bright from all directions. For a flat specular (mirrorlike) surface, the BRDF is an impulse function as determined by the laws of reflection.

The scene radiance at a point on the surface of an object depends on the reflectance properties of the surface, as well as on the intensity and direction of the illumination sources. For example, for a Lambertian surface illuminated by a point light source, the scene radiance is proportional to the cosine of the angle between the surface normal and the direction of illumination. This relationship between surface orientation and brightness is captured in the *reflectance map*. In the reflectance map for a given surface and illumination, contours of equal brightness are plotted as a function of surface orientation specified by the gradient space coordinates (p, q), where p and q are the surface gradients in the x and y directions, respectively. A typical reflectance map for a Lambertian surface illuminated by a point source is shown in Fig. 19.2. In this figure, the brightest point corresponds to the surface orientation such that its normal points in the direction of the source. Since image brightness is proportional to scene radiance, a direct relationship exists between the image intensity at an image point and the orientation of the surface of the corresponding scene point. *Shape-from-shading* algorithms exploit this relationship to recover the three-dimensional object shape. *Photometric stereo* exploits the same principles to recover the shape of objects from multiple images obtained by illuminating the scene from different directions.

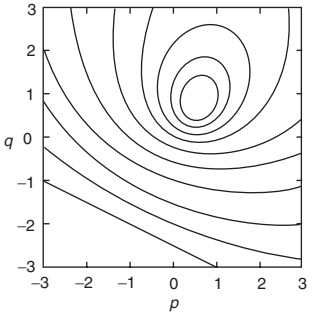


FIGURE 19.2 The reflectance map for a typical Lambertian surface illuminated by a point source from the direction $(p, q) = (0.2, 0.4)$.

Sampling and Quantization

A continuous function cannot be represented exactly in a computer. The interface between the imaging system and the computer must sample the continuous image function at a finite number of points and represent each sample within a finite number of bits. This is called **sampling** and **quantization**. Each image sample is a *pixel*. Generally images are sampled on a regular grid of squares so that the horizontal and vertical distances between pixels are the same throughout the image. Each pixel is represented in the computer as a small integer and represented as a two-dimensional array. Frequently, a pixel is represented

© 2006 by Taylor & Francis Group, LLC

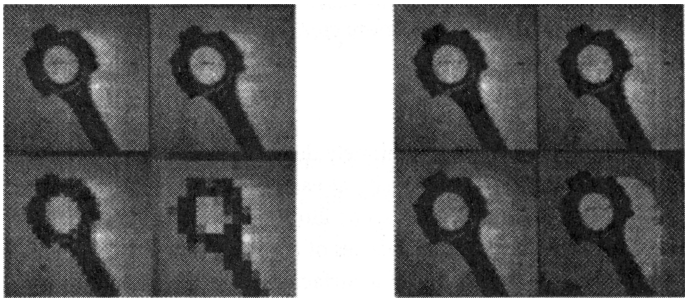


FIGURE 19.3 Sample image: (a) various spatial resolutions, (b) at differing numbers of quantization levels. Notice the blocky structure in (a) and the appearance of many *false contours* in (b). (*Source:* Jain, R., Kasturi, R., and Schunk, B.G. 1995. *Machine Vision.* McGraw-Hill, New York. With permission.)

as an unsigned 8-b integer in the range of $(0, 255)$, with 0 corresponding to black, 255 corresponding to white, and shades of gray distributed in between.

Many cameras acquire an analog image, which is then sampled and quantized to convert it to a digital image. The sampling rate determines how many pixels the digital image will have (image resolution), and the quantization determines how many intensity levels will be used to represent the intensity value at each sample point. As shown in Fig. 19.3(a) and Fig. 19.3(b), an image looks very different at different sampling rates and quantization levels. In most machine vision applications, the sampling rate and quantization are predetermined due to the limited choice available of cameras and image acquisition hardware. In many applications, however, it is important to know the effects of sampling and quantization.

Color Vision

The previous sections were only concerned with the intensity of light; however, light consists of a full spectrum of wavelengths. Images can include samples from a number of different wavelengths, leading to a color image. This perception of color depends on (1) the spectral reflectance of scene surfaces, (2) the spectral content of the illuminating light source, and (3) the spectral response of the sensors in the imaging system. In humans, color perception is attributed to differences in the spectral responses of the millions of neurochemical photoreceptors—rods and cones—in the retina. The rods are responsible for monochromatic vision and the cones are responsible for the sensation of color. The human visual system consists of three types of cones, each with a different spectral response. The total response each of the cones is often modeled by the integral

$$R_i(\lambda) = \int f(\lambda) r(\lambda) h_i(\lambda) d\lambda \qquad i = 1, 2, 3 \tag{19.2}$$

where λ is the wavelength of light incident on the receptor, $f(\lambda)$ the spectral composition of the illumination, $r(\lambda)$ the spectral reflectance function of the reflecting surface, and $h_i(\lambda)$ the spectral response of the ith type of receptor (cone).

In machine vision, color images are typically represented by the output of three different sensors, each having a different spectral response (e.g., the primary colors red, green, and blue). The choice of these primary colors determines the range of possible colors that can be realized by weighted combinations of the primaries. However, it is not necessary for the response of the sensors to be restricted to visible light. The sensors could just as easily respond to wavelengths in the infrared, ultraviolet, or X-ray portions of the electromagnetic spectrum, for example. In either case, the imaging system creates multiple images, called *channels*, one for each sensor. If the outputs from three sensors are used, this translates into a threefold increase in the processing and memory requirements. However, this increase is often offset by exploiting the extra dimensionality that is now available.

© 2006 by Taylor & Francis Group, LLC

Range Imaging

Although three-dimensional information can be extracted from images with two-dimensional intensity—using image cues such as shading, texture, and motion—the problem is greatly simplified by range imaging. Range imaging is aquiring images for which the value at each pixel is a function of the distance of the corresponding point of the object from the sensor. The resulting images are known as *range images* or *depth maps*. A sample range image of a coffee cup is shown in Fig.19.4. Here we will briefly discuss two common range imaging techniques, namely *imaging radar* and *triangulation*.

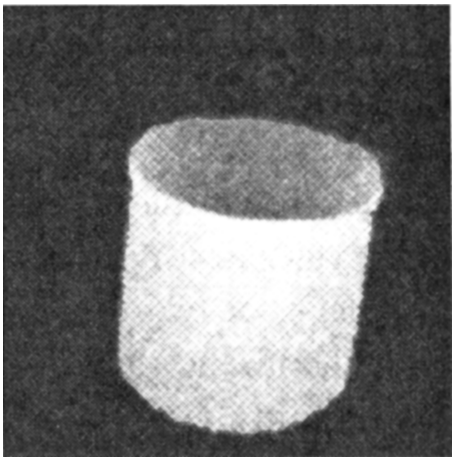


FIGURE 19.4 A range image of a coffee mug.

Imaging Radar

In a time-of-flight pulsed radar system, the distance to the object is computed by observing the time difference between transmitted and received electromagnetic pulses. Range information can also be obtained by detecting the phase difference between the transmitted and received waves of an amplitude-modulated beam or by detecting the beat frequency in a coherently mixed transmitted-and-received signal in a frequency-modulated beam.

Triangulation

In an active triangulation-based range imaging system, a light projector and camera are aligned along the z axis separated by a baseline distance b as shown in Fig. 19.5. The object coordinates (x, y, z) are related to the image coordinates (x', y') and the projection angle θ by the following equation:

$$[x\ y\ z] = \frac{b}{f\cot\theta - x'}[x'y'f] \tag{19.3}$$

The accuracy with which the angle θ and the horizontal position x' can be measured determines the range resolution of such a triangulation system. This system of active triangulation uses a single point light source that is swept across the scene in a raster scan fashion.

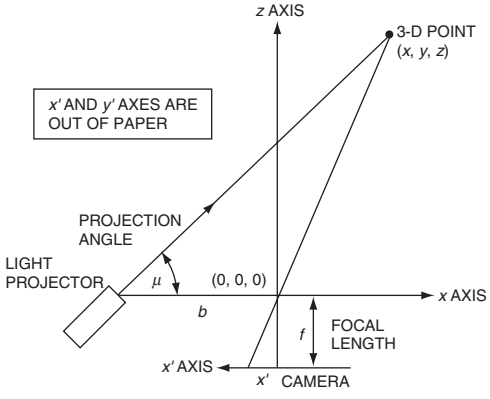


FIGURE 19.5 A triangulation-based range imaging system. (*Source:* Best, P.J. 1988. Active, optical range imaging sensor. *Machine Vision and Applications* 1(2): 127–152.)

© 2006 by Taylor & Francis Group, LLC

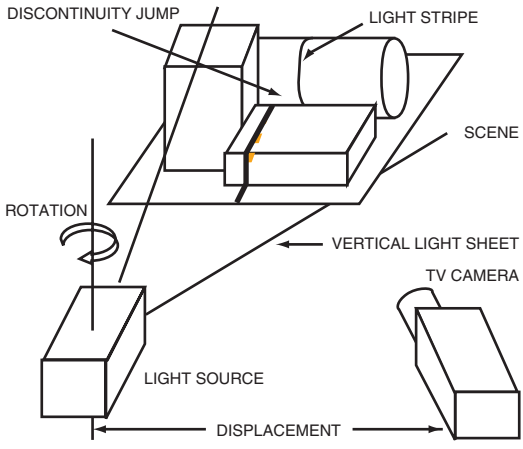


FIGURE 19.6 A typical structured lighting system. (*Source:* Jorvis, R.A. 1983. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5(2):122–139. ©1983 IEEE.)

Structured Lighting

Imaging using structured lighting refers to systems in which the scene is illuminated by a known geometric pattern of light. In a simple point projection system—like the triangulation system already described—the scene is illuminated, one point at a time, in a two-dimensional grid pattern. The depth at each point is calculated using the previous equation to obtain the range image. In a typical structured lighting system, either planes or two-dimensional patterns of light are projected onto the scene. A camera, which is displaced spatially from the source of illumination, records the patterns of light projected on the object light surfaces. The distortions contained in the observed images of the patterns are determined by the shape and orientation of surfaces of the objects on which the patterns are projected (see Fig. 19.6).

In situations where the scene is changing dynamically, projecting single light stripes in sequence may not be practical due to the sequential nature of the system. In this situation, a set of multiple light stripes—each uniquely encoded—is projected. For example, using a *binary-coded* scheme, a complete set of data can be acquired by projecting only $\log_2 N$ patterns, where $N - 1$ is the total number of stripes in each pattern. Other structured lighting techniques include *color coding*, *grid coding*, and *Fourier domain processing*. The primary drawback of any structured lighting technique is that data cannot be obtained for object points that are not visible to either the light source or the imaging camera.

Active Vision

Most machine vision systems rely on data captured by hardware with fixed characteristics. These systems include passive sensing systems—such as video cameras—and active sensing systems—such as laser range finders. In an active vision system, the parameters and characteristics of data capture are dynamically controlled by the scene interpretation system. Active vision systems may employ either active or passive sensors. In an active vision system, however, the state parameters of the sensors, such as focus, aperature, vergence, and illumination, are controlled to acquire data that will facilitate scene interpretation.

19.3 Segmentation

An image must be analyzed and its relevant features extracted before more abstract representations and descriptions can be generated. Careful selection of these so-called low-level operations is critical for the success of higher level scene interpretation algorithms. One of the first operations that a machine vi-

© 2006 by Taylor & Francis Group, LLC

sion system must perform is the separation of objects from the background. This operation, commonly called **segmentation**, is approached in one of two ways: (1) an *edge-based* method for locating discontinuities in certain properties or (2) a *region-based* method for the grouping of pixels according to certain similarities.

Edge-Based Segmentation

In an edge-based approach, the boundaries of objects are used to partition an image. Points that lie on the boundaries of an object must be marked. Such points, called *edge points*, can often be detected by analyzing the local neighborhood of a point. By definition, the regions on either side of an edge point (i.e., the object and background) have dissimilar characteristics. Thus, in edge detection, the emphasis is on detecting dissimilarities in the neighborhoods of points.

Gradient

An *edge* in an image is indicated by a significant local change in image intensity, usually associated with a discontinuity in either the image intensity or the first derivative of the image intensity. Therefore, a direct approach to edge detection is to compute the *gradient* at each pixel in an image. The gradient is the two-dimensional equivalent to the first derivative and is defined as the vector

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{df}{dx} \\ \dfrac{df}{dy} \end{bmatrix} \tag{19.4}$$

where $f(x,y)$ represents the two-dimensional image intensity function. The magnitude and direction of the gradient in terms of the first partial derivatives G_x and G_y are given by

$$G(x,y) = \sqrt{G_x^2 + G_y^2} \quad \text{and} \quad \alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{19.5}$$

Only those pixels whose gradient magnitude is larger than a predefined threshold are identified as edge pixels.

Many approximations have been used to compute the partial derivatives in the x and y directions. One of the earlier edge detectors, the *Roberts' cross operator*, computes the partial derivatives by taking the difference in intensities of the diagonal pixels in a 2×2 window. Other approximations, such at the *Prewitt* and *Sobel* operators, use a 3×3 window to calculate the partial derivatives. Occasionally, operators that compute the edge strength in a number of specific directions are used in machine vision. These directional operators are defined over 3×3 or larger neighborhoods. For applications where such simple edge detectors are not adequate, the *Canny* edge detector, which attempts to optimize noise suppression with edge localization, is often used. See Fig. 19.7 for a description of the window operators used in each of the mentioned edge operators.

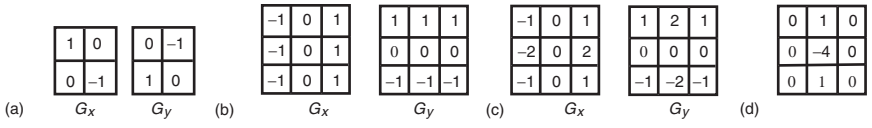


FIGURE 19.7 Operators: (a) the Roberts' operators, (b) Prewitt operators, (c) Sobel operators, and (d) the Laplacian operator. Note that the Laplacian operator is *not* a directional operator.

© 2006 by Taylor & Francis Group, LLC

Laplacian

When using gradient-based edge detectors, multiple responses are often obtained at adjacent pixels depending on the rate of transition of the edge. After **thresholding**, thick edges are usually obtained. However, it is desired to obtain only a single response for a single edge; in other words, only keep the maximum of the gradient and not every response that is above a specified threshold. The maximum of the gradient is found by detecting the zero crossings in the second derivative of the image intensity function. The *Laplacian* is defined in terms of the second partial derivatives as follows

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{19.6}$$

The second partial derivatives along the x and y directions are approximated using the difference equations

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) - 2f(x, y) + f(x-1, y)$$

and

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1) \tag{19.7}$$

By combining these two equations, the Laplacian can be implemented in a single window operator (see Fig. 19.7).

The response of the Laplacian operator can be computed in one step by convolving the image with the appropriate window operator. Nevertheless, since the Laplacian is an approximation to the second derivative, it reacts more strongly to lines, line ends, and noise pixels than it does to edges; plus it generates two responses for each edge, one positive and one negative on either side of the edge. Hence, the zero crossing between the two responses is used to localize the edge.

Laplacian of Gaussian

Detecting edge points from the zero crossings of the second derivative of the image intensity is very noisy. Therefore, it is often desirable to filter out the noise before edge detection. To do this, the *Laplacian of Gaussian* (LoG) combines Gaussian filtering with the Laplacian for edge detection. The output of the LoG operator, $h(x, y)$, is obtained by the convolution operation

$$h(x, y) = \nabla^2(g(x, y) * f(x, y)) \tag{19.8}$$

where

$$g(x, y) = e^{-\left[\frac{(x^2+y^2)}{2\sigma^2}\right]} \tag{19.9}$$

is the equation for the Gaussian filter (neglecting the constant factor). Using the derivative rule for convolution, we get the equivalent expression

$$h(x, y) = [\nabla^2 g(x, y)] * f(x, y) \tag{19.10}$$

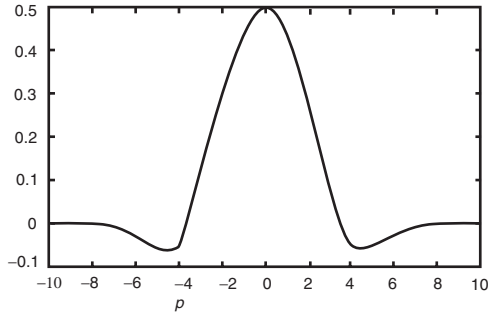


FIGURE 19.8 The *inverted* Laplacian of Gaussian function.

where the term

$$\nabla^2 g(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right) e^{-\left[\frac{(x^2+y^2)}{2\sigma^2}\right]} \tag{19.11}$$

is commonly called the *Mexican hat* operator because of its appearance when plotted (see Fig. 19.8).

This result shows that the Laplacian of Gaussian can be obtained in either of two ways: (1) Convolve the image with a Gaussian smoothing filter and compute the Laplacian of the result. (2) Convolve the image with a filter that is the Laplacian of the Gaussian filter. The zero crossings still must be detected in order

© 2006 by Taylor & Francis Group, LLC

to localize the edges; and like the Laplacian, only those edge points whose corresponding first derivative is above a specified threshold are marked as edge points.

When using the Laplacian of Gaussian approach, edges are detected at a particular resolution (scale) depending on the spread of the Gaussian filter used. The determination of real edges in any image may require the combination of information from operators at several scales using a *scale-space* approach. The presence of edges at a particular resolution are found at the appropriate scale, and the exact locations of the edges are obtained by tracking their locations through lower values of σ.

Surface Fitting

Since a digital image is actually a sampling of a continuous function of two-dimensional spatial variables, another viable approach to edge detection is to approximate and reconstruct the underlying continuous spatial function that the image represents. This process is termed *surface fitting*. The intensity values in the neighborhood of a point can be used to obtain the underlying continuous intensity surface, which then can be used as a best approximation from which properties in the neighborhood of the point can be computed. The continuous intensity finction can be described by

$$z = f(x, y) \tag{19.12}$$

If the image represents one surface, the preceeding equation will correctly characterize the image. However, an image generally contains several surfaces, in which case this equation would be satisfied only at local surface points. Therefore, this approach uses the intensity values in the neighborhood about each pixel to approximate a local surface about that point. The *facet model* is a result of this idea. In the facet model, the neighborhood of a point is approximated by the cubic surface patch that best fits the area using

$$f(r, c) = k_1 + k_2 r + k_3 c + k_4 r^2 + k_5 r c + k_6 c^2 + k_7 r^3 + k_8 r^2 c + k_9 r c^2 + k_{10} c^3 \tag{19.13}$$

where r and c are local coordinates about the point in the center of the neighborhood being approximated. From this approximation, the second directional derivative can be calculated and the zero crossings can be detected, and the edge points can be found.

Edge Linking

The detection of edge pixels is only part of the segmentation task. The outputs of typical edge detectors must be linked to form complete boundaries of objects. Edge pixels seldom from closed boundaries; missed edge points will result in breaks in the boundaries. Thus, *edge linking* is an extremely important, but difficult, step in image segmentation. Several methods have been suggested to improve the performance of edge detectors ranging from *relaxation-based edge linking*, which uses the magnitude and direction of an edge point to locate other edges in the neighborhood, to *sequential edge detectors*—also called *edge trackers*—which finds a strong edge point and grows the object boundary by considering neighboring edge strengths and directions.

Region-Based Segmentation

In region-based segmentation, all pixels corresponding to a single object are grouped together and are marked to indicate that they belong to the same object. The grouping of the points is based on some criterion that distinguishes pixels belonging to the same object from all other pixels. Points that have similar characteristics are identified and grouped into a set of connected points, called *regions*. Such points usually belong to a single object or part of an object. Several techniques are available for segmenting an image using this region-based approach.

Region Formation

The segmentation process usually begins with a simple region formation step. In this step, intrinsic characteristics of the image are used to form initial regions. Thresholds obtained from a **histogram** of the image intensity are commonly used to perform this initial grouping. In general, an image will have several

© 2006 by Taylor & Francis Group, LLC

regions, each of which may have different intrinsic characteristics. In such cases, the intensity histogram of the image will show several peaks; each peak may correspond to one or more regions. Several thresholds are selected using these peaks. After thresholding, a connected-component algorithm can be used to find initial regions.

Thresholding techniques usually produce too many regions. Since they were formed based only on first-order characteristics, the regions obtained are usually simplistic and do not correspond to complete objects. The regions obtained via thresholding may be considered to be only the first step in segmentation. After the initial histogram-based segmentation, more sophisticated techniques are used to refine the segmentation.

Split and Merge

Automatic refinement of an initial intensity-based segmentation is often done using a combination of *split* and *merge* operations. Split and merge operations eliminate false boundaries and spurious regions by either splitting a region that contains pieces from more than one object or merging adjacent regions that actually belong to the same object. If some property of a region is not uniform, the region should be split. Segmentation based on the split approach starts with large regions. In many cases, the whole image may be used as the starting region. Several decisions must be made before a region is split. One is to decide "when" a property is nonuniform over a region; another is "how" to split a region so that the property for each of the resulting components is uniform. These decisions usually are not easy to make. In some applications, the variance of the intensity values is used as a measure of uniformity.

More difficult than determining the property uniformity is deciding "where" to split a region. Splitting regions based on property values is very difficult. One approach used when trying to determine the best boundaries with which to divide a region is to consider edgeness values within a region. The easiest schemes for splitting region are those that divide a region into a fixed number of equal regions; such methods are called *regular decomposition methods*. For example, in the *quad tree* approach, if a region is considered nonuniform, it is split into four equal quadrants in each step.

Many approaches have been proposed to judge the similarity of regions. Broadly, the approaches are based on either the characteristics of regions or the weakness of edges between them. Two common approaches to judging the similarity of adjacent regions based on region characteristics are to (1) compare their mean intensities and (2) assume their intensity values are drawn from a known probability distribution. In the first approach, if their mean intensities do not differ by more than some predetermined value, the regions are considered to be similar and are candidates for merging. A modified form of this approach uses surface fitting to determine if the regions can be approximated by one surface. In the latter approach, the decision of whether or not to merge adjacent regions is based on considering the probability that they will have the same statistical distribution of intensity values. This approach uses hypothesis testing to judge the similarity of adjacent regions.

Another approach to merging is to combine two regions only if the boundary between them is weak. A weak boundary is one for which the intensities on either side differ by less than a given threshold. This approach attempts to remove weak edges between adjacent regions by considering not only the intensity characteristics, but also the length of the common boundary. The common boundary is dissolved if it is weak and the resulting boundary (of the merged region) does not grow too fast.

Split and merge operations may be used together. After a presegmentation based on thresholding, a succession of splits and merges may be applied as dictated by the properties of the regions. Such schemes have been proposed for segmentation of complex scenes. Domain knowledge with which the split and merge operations can be controlled may also be introduced.

Other Segmentation Methods

In the preceeding discussion, the primary focus was with intensity images. Segmentation techniques that are based on *color*, *texture*, and *motion* have also been developed. Segmentation based on spectral pattern classification techniques is used extensively in remote-sensing applications.

© 2006 by Taylor & Francis Group, LLC

19.4 Feature Extraction and Matching

Segmented images are often represented in a compact form to facilitate further abstraction. Representation schemes are chosen to match methods used for object recognition and description. The task of object recognition requires matching an object description in an image to models of known objects. The models, in turn, use certain descriptive features and their relations. Matching also plays an important role in other aspects of information recovery from images. In the following, schemes for representation and description that are popular in machine vision are discussed along with techniques for feature extraction and matching.

Representation and Description

Representation and description of symbolic information can be approached in many ways. One approach is to represent the object in terms of its bounding curve. Popular among several methods developed for boundary representation are chain codes, **polygonalization**, one-dimensional signatures, and representations using dominant points for representing an object. Another approach is to obtain region-based shape descriptors, such as topological or texture descriptors. Representation and shape description schemes are usually chosen such that the descriptors are invariant to rotation, translation, and scale changes.

Chain Codes

One of the earliest methods for representing a boundary uses directional codes, called *chain codes*. The object boundary is resampled at an appropriate scale, and an ordered list of points along the boundary is represented by a string of directions codes (see Fig. 19.9(a)). Often to retain all of the information in the boundary, the resampling step is bypassed. However, resampling eliminates minor fluctuations that typically are due to noise. Chain codes possess some attractive features; namely, the rotation of an object by multiples of 45° is easily implemented; the derivative of the chain code, the *difference code* obtained by computing the first differences of the chain code, is rotation invariant; and other characteristics of a region, such as area and corners, can be computed directly from the chain code. The limitations of this representation method are attributed to the restricted directions used to represent the tangent at a point. Although codes with larger numbers of directions are occasionally used, the eight-directional chain code is the most commonly used.

Polygonalization

Polygonal approximation of boundaries has been studied extensively and numerous methods have been developed. The fit of a polygon is made to reduce a chosen error criterion between the approximation and the original curve. In an *interative endpoint* scheme, the first step is to connect a straight-line segment between the two farthest points on the boundary. The perpendicular distances from the segment to each point on the curve is measured. If any distance is greater than a chosen threshold, the segment is replaced

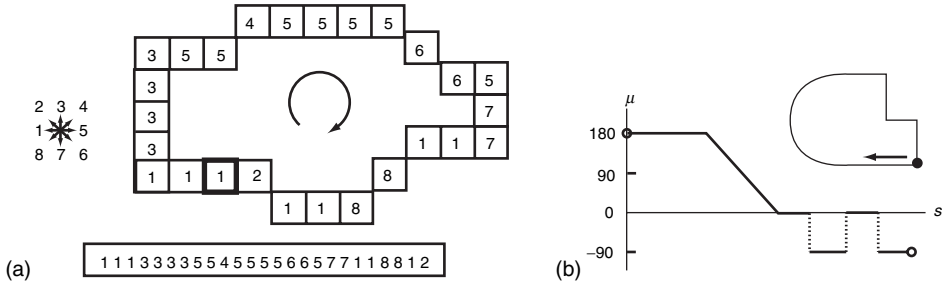


FIGURE 19.9 (a) An object and its chain code. (b) A simple geometric object and its one-dimensional (s,θ) signature.

© 2006 by Taylor & Francis Group, LLC

by two segments; one each from the segment endpoint to the curve point where the distance to the segment is greatest. In another approach, a straight-line fit is constrained to pass within a radius around each data point. The line segment is grown from the first point, and when further extension of the line segment causes it to fall outside the radius point, a new line is started. A *minimax* approach can also be used in which the line segment approximations are chosen to minimize the maximum distance between the data points and the approximating line segment.

Besides polygonal approximation methods, higher order curve- and spline-fitting methods may be used where more precise approximations are required. These are more computationally expensive than most polygonalization methods, and they can be more difficult to apply.

One-Dimensional Signatures

The slope of the tangent to the curve denoted by the angle θ as a function of s, the position along the curve from an arbitrary starting point, is used to represent curves in many applications. Plots of these functions have some interesting and useful characteristics. Horizontal lines in s–θ plots represent straight lines and straight lines at an angle with the horizontal represent circular arcs whose radii are proportional to the slope of the straight line. An s–θ curve can also be treated as a periodic function with a period given by the perimeter of the curve; hence, Fourier techniques can be used. Other functions, such as the distance to the curve from an arbitrary point inside the curve plotted as a function of the angle with reference to the horizontal, are also used as shape signatures. Figure 19.9(b) shows a simple geometric object and its one-dimensional signature.

Boundary Descriptors

Descriptors for objects represented by their boundaries may be generated using the representations already described. Simple descriptors, such as perimeter, length, and orientation of the major axis, shape number, and eccentricity, may be readily computed from the boundary data. The ordered set of points on the boundary having two-dimensional coordinates (x_k, y_k), where $k = 1, \ldots, N$ and N is the total number of points on the boundary, can be treated as a one-dimensional complex function $x_k + iy_k$. The coefficients of the discrete Fourier transform applied to this function can also be used as a shape descriptor.

Other descriptors that use the one-dimensional signature functions may also be obtained. A major drawback of many of these descriptors is that complete data for the object boundary are required. Because of either problems in segmentation or occlusions in the image, however, complete data for the object boundary are often not available. In such instances, recognition strageties based on partial information are needed.

Many region-based representation and description methods analogous to those used to represent object boundaries have been developed. Examples of such methods are the *medial-axis transform*, *skeletons*, *convex hulls*, and *deficiencies*. *Morphological operators* have been developed to extract shape features and generate useful descriptions of object shape. Topological descriptors, such as *Euler number*, are also useful to characterize shape. The facet model has even been used to generate a topographic primal sketch of images using a set of seven descriptive labels, including pit, peak, and ridge to name a few. Such descriptions are useful in object matching. Region-based representation and description are particularly useful to describe objects for which properties within the region, for example, texture, are significant for object recognition. For a complete discussion of these topics, refer to the references at the end of this chapter.

Feature Extraction

If the object to be recognized has unique discriminating features, special-purpose algorithms may be employed to extract such features. Important for object matching and recognition are corners, high-curvature regions, inflection points, or other places along curves at which curvature discontinuities exist. In region-based matching methods, identifying groups of pixels that can be easily distinguished and identified is important. Many methods have been proposed to detect both dominant points in curves and interesting points in regions.

© 2006 by Taylor & Francis Group, LLC

Critical Points

The detection of *critical points*, also called *dominant points*, in curves, such as corners and inflection points, is important for subsequent object matching and recognition. Most algorithms for detecting critical points mark local curvature maxima as dominant. One approach is to analyze the deviations of a curve from a chord to detect dominant points along the curve. Points are marked as either being critical or belonging to a smooth or noisy interval. These markings depend on whether the curve makes a single excursion away from the chord, stays close to the chord, or makes two or more excursions away from the chord, respectively. Other approaches use mathematical expressions to directly compute and plot the curvature values along the contour.

Interesting Points

Points used in matching between two images must be ones that are easily identified and matched. These are known as *interesting points*. Obviously, those points in a uniform region and on edges are not good candidates for matching. Interest operators find image areas with high variance. In applications such as stereo and structure from motion, images should have enough such interesting regions to facilitate matching.

One commonly used interest operator, the *Moravec interest operator*, uses directional variances as a measure of how interesting a point is. A point is considered interesting if it has a local maximum of minimal sums of directional variances. The directional variances in the local neighborhood about a point are calculated by

$$\begin{aligned} I_1 &= \sum_{(x,y)\in s} [f(x,y) - f(x,y+1)]^2 \\ I_2 &= \sum_{(x,y)\in s} [f(x,y) - f(x+1,y)]^2 \\ I_3 &= \sum_{(x,y)\in s} [f(x,y) - f(x+1,y+1)]^2 \\ I_4 &= \sum_{(x,y)\in s} [f(x,y) - f(x+1,y-1)]^2 \end{aligned} \tag{19.14}$$

where s represents the neighborhood about the current point. Typical neighborhoods range from 5×5 to 11×11 pixels. The *interestingness* of the point is then given by

$$I(x,y) = \min(I_1, I_2, I_3, I_4) \tag{19.15}$$

Feature points are chosen where the interestingness is a local maximum. Doing this eliminates the detection of simple edge points since they have no variance in the direction of the edge. Furthermore, a point is considered a good interesting point if its local maximum is above a preset threshold. The Moravec interest operator has found extensive use in stereo matching applications.

Matching

Matching plays a very important role in many phases of machine vision. Object recognition requires matching a description of an object with models of known objects. The goal of matching is to either (1) detect the presence of a known entity, object, or feature or (2) find what an unknown image component is. The difficulty in achieving these matching goals is first encountered with goal-directed matching, wherein the goal is to find a very specific entity in an image. Usually, the location of all instances of such an entity must be found. In stereo and structure-from-motion applications, entities are obtained in one image, and their locations are then determined in a second image. The second problem requires matching an unknown entity with several models to determine which model matches best.

© 2006 by Taylor & Francis Group, LLC

Point Pattern Matching

In matching points in two slightly different images of the same scene (i.e., in a stereo image pair or a motion sequence), interesting points are detected by applying an operator such as the Moravec interest operator. The correspondence process considers the local structure of a selected point in one image in order to assign initial matchable candidate points from the second image. However, this *correspondence problem* is not an easy one to solve, and many constraints are often imposed to ease the process. For example, in stereo-matching applications, the displacements of a point from one image to the other are usually small. Thus, only points within a local neighborhood are considered for matching. To obtain the final correspondence, the set of initial matches is refined by computing the similarity in global structure around each candidate point. In dynamic-scene analysis, one may assume that motions of neighboring points do not differ significantly. To obtain the final matching of points in the two images, relaxation techniques are often employed. An example of the point matching problem is illustrated in Fig. 19.10.

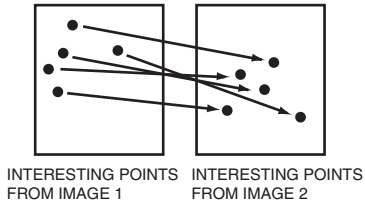


FIGURE 19.10 An illustration of the point pattern matching process.

Template Matching

In some applications, a particular pictorial or iconic structure, called a *template*, is to be detected in an image. Templates are usually represented by small two-dimensional intensity functions (typically less than 64×64 pixels). *Template matching* is the process of moving the template over the entire image and detecting the locations at which the template best fits the image. A common measure of similarity to determine a match is the *normalized cross correlation*. The correlation coefficient is given by

$$M(x,y) = \frac{\sum\sum_{(u,v)\in R} g(u,v) f(x+u, y+v)}{\left\{\sum\sum_{(u,v)\in R} f^2(x+u, y+v)\right\}^{1/2}} \tag{19.16}$$

where $g(u,v)$ is the template, $f(x,y)$ the image, and R the region spanned by the template. The value M takes maximum value for the point (x,y) at which $g = cf$ (in other words where the pixel values of the template and the image only differ by a constant scaling factor). By thresholding the result of this cross-correlation operation, the locations of a template match can be found. Figure 19.11 shows a test image, a template, and the results of the normalized cross correlation.

A major limitation of the template-matching technique is its sensitivity to the rotation and scaling of objects. To match scaled and rotated objects, separate templates must be constructed. In some approaches, a template is partitioned into several subtemplates, and matching is carried out for these subtemplates. The relationships among the subtemplates are verified in the final matching step.

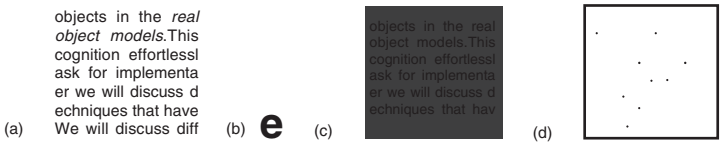


FIGURE 19.11 (a) A test image. (b) a template (enlarged) of the letter *e*. (c) the results of the normalized cross correlation. (d) the thresholded result indicating the match locations ($T = 240$). (*Source:* Jain, R., Kasturi, R., and Schunck, B.G. 1995. *Machine Vision.* McGraw-Hill, New York. With Permission.)

© 2006 by Taylor & Francis Group, LLC

Hough Transform

Parametric transforms, such as the Hough transform, are useful methods for the recognition of straight lines and curves. For a straight line, the transformation into the parameter space is defined by the parametric representation of a straight line given by

$$\rho = x\cos\theta + y\sin\theta \tag{19.17}$$

where (ρ, θ) are the variables in the parameter space that represent the length and orientation, respectively, of the normal to the line from the origin. Each point (x, y) in the image plane transforms into a sinusoid in the (ρ, θ) domain. However, all sinusoids corresponding to points that are colinear in the image plane will intersect at a single point in the Hough domain. Thus, pixels belonging to straight lines can be easily detected.

The Hough transform can also be defined to recognize other types of curves. For example, points on a circle can be detected by searching through a three-dimensional parameter space of (x_c, y_c, r) where the first two parameters define the location of the center of the circle and r represents the radius. The Hough transform can also be generalized to detect arbitrary shapes. However, one problem with the Hough transform is the large parameter search space required to detect even simple curves. This problem can be alleviated somewhat by the use of additional information that may be available in the spatial domain. For example, in detecting circles by the brute-force method, searching must be done in a three-dimensional parameter space; however, when the direction of the curve gradient is known, the search is restricted to only one dimension.

Pattern Classification

Features extracted from images may be represented in a multidimensional feature space. A classifier—such as a minimum distance classifier, which is used extensively in statistical pattern recognition—may be used to identify objects. This pattern classification method is especially useful in applications wherein the objective is to assign a label, from among many possible labels, to a region in the image. The classifier may be taught in a supervised-learning mode, by using prototypes of known object classes, or in an unsupervised mode, in which the learning is automatic. Proper selection of features is crucial for the success of this approach. Several methods of pattern classification using structural relationships have also been developed. For a complete discussion of pattern classification, see Duda and Hart (1973).

19.5 Three-Dimensional Object Recognition

The real world that we see and touch is primarily composed of three-dimensional solid objects. When an object is viewed for the first time, people typically gather information about that object from many different viewpoints. The process of gathering detailed object information and storing that information is referred to as model formation. Once a person is familiar with many objects, the objects are then identified from an arbitrary viewpoint without further investigation. People are also able to identify, locate, and qualitatively describe the orientation of objects in black-and-white photographs. This basic capability is significant to machine vision because it involves the spatial variation of a single parameter within a framed rectangular region that corresponds to a fixed, single view of the world.

Recognition System Components

Recognition implies awareness of something already known. In modeling real-world objects for recognition purposes, different kinds of schemes have been used. To determine how recognition will take place, a method for matching model data to sensor data must be considered. A straightforward blind-search approach would entail (1) transforming all possible combinations of all possible known object models in all possible distinguishable orientations and locations into a digitized sensor format and (2) matching based on the minimization of a matching-error criterion. Clearly, this approach is impractical. On the other hand, since object models contain more object information than sensor data, we are prohibited from

© 2006 by Taylor & Francis Group, LLC

transforming sensor data into complete model data and matching in the model data format. However, this does not prevent us from matching with partial model data. As a result, working with an intermediate domain that is computable from both sensor and model data is advantageous. This domain is referred to as the symbolic scene description domain. A matching procedure is carried out on the quantities in this domain, which are referred to as *features*.

The interactions between the individual components of a recognition system are illustrated in Fig. 19.12. The image formation process creates intensity or range data based purely on physical principles. The description process acts on the sensor data and extracts relevant application-independent features. This process is completely data-driven and includes only the knowledge of the image formation process. The modeling process provides object models for real-world objects. Object reconstruction from sensor data is one method for building models automatically. The understanding, or recognition, process involves an algorithm to perform matching between model and data descriptions. This process might include data- and model-driven subprocesses, where segmented sensor data regions seek explanations in terms of models and hypothesized models seek verification from the data. The rendering process produces synthetic sensor data from object models. Rendering provides an important feedback link by allowing an autonomous system to check on its own understanding of the sensor data by comparing synthetic images to sensed images.

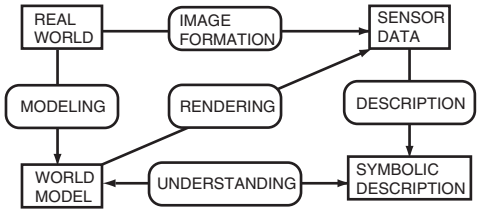


FIGURE 19.12 The components of an object recognition system. (*Source:* Best, P. and Jain, R.C. 1985. Three dimensional object recognition. *ACM Computing Surveys* 17(1).)

Object Representation Schemes

Numerous schemes have been developed for representing three-dimensional objects. The choice of a particular scheme is governed by its intended application. In computer graphics, schemes such as the wire-frame and constructive solid-geometry representations are popular since their data structures are suitable for image rendering. In machine vision systems, other methods such as generalized cones and characteristic views are used extensively. In the following, we briefly describe some commonly used object representation schemes.

Wire Frame

The wire-frame representation of a three-dimensional object consists of a three-dimensional vertex point list and an edge list of vertex pairs. Although this representation is very simple, it is an ambiguous representation for determining such quantities as surface area and volume of an object. Wire-frame models can sometimes be interpretted as several different solid objects or as different orientations of the same object. Figure 19.13(a) shows the wire-frame representation of a simple three-dimensional object.

Constructive Solid Geometry

The constructive solid-geometry (CSG) representations of an object is specified in terms of a set of three-dimensional volumetric primitives (blocks, cylinders, cones, spheres, etc.) and a set of boolean operators: union, intersection, and difference. Figure 19.13(b) shows the CSG representation for a simple geometric object. The storage data structure is a binary tree, where the terminal nodes are instances of the geometric primitives and the branching nodes represent the Boolean set operations and positioning information. CSG trees define object surface area unambiguously and can, with very little data, represent complex objects. However, the boundary evaluation algorithms required to obtain usable surface information are very computationally expensive. Also, general sculptured surfaces are not easily represented using CSG models.

© 2006 by Taylor & Francis Group, LLC

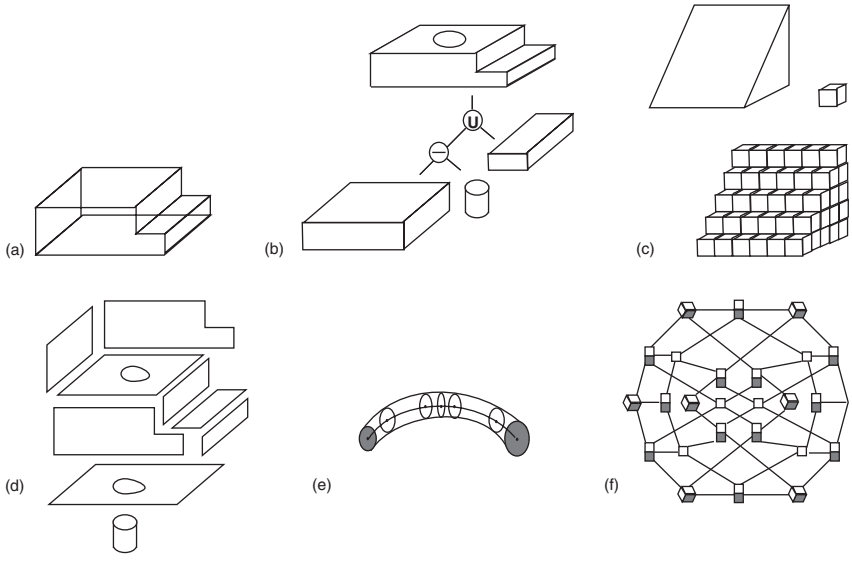


FIGURE 19.13 Various object representation schemes: (a) wire-frame, (b) constructive solid geometry, (c) spatial occupancy, (d) surface-boundary, (e) generalized-cone, (f) aspect graph.

Spatial Occupancy

Spatial-occupancy representations use nonoverlapping subregions of three-dimensional space occupied by an object to define that object. This method unambiguously defines an object's volume. Some commonly used single primitive representations of this type are the voxel and octree representations. *Voxels* are small volume elements of discretized three-dimensional space (see Fig. 19.13(c)). They are usually fixed-size cubes. Objects are represented by the lists of voxels occupied by the object. Voxel representations tend to be very memory intensive, but algorithms using them tend to be very simple. An *octree* is a hierarchical representation of spatial occupancy. Volumes are decomposed into cubes of different size, where the cube size depends on the distance from the root node. Each branching node of the tree represents a cube and points to eight other nodes, each of which describes object volume occupancy in the corresponding octant subcubes of the branching node cube. The octree representation offers the advantages of the voxel description but is more compact. Beacuse of this compactness, more complicated algorithms are required for many computations than those needed for voxel representation.

Surface Boundary

The surface-boundary representation defines a solid object by defining the three-dimensional surfaces that bound the object. Figure 19.13(d) shows a surface-boundary representation for a simple geometric object. The simplest boundary representation is the triangle-faced polyhedron, which can be stored as a list of three-dimensional triangles. Arbitrary surfaces are approximated to any desired degree of accuracy by utilizing many triangles. A slightly more compact representation allows the replacement of adjacent, connected, coplanar triangles with arbitrary n-sided planar polygons. Structural relationships between bounding surfaces may also be included as part of the model.

Generalized Cone

In the generalized-cone (generalized-cylinder or sweep) representation, an object is represented by a three-dimensional space curve that acts as a spine or axis of the cone, a two-dimensional cross-sectional figure, and a sweeping rule that defines how the cross section is to be swept (and possibly modified) along the space curve (see Fig. 19.13(e)). Generalized cones are well suited for representing many real-world shapes. However, certain objects, such as the human face or an automobile body, are difficult to represent as generalized cones. Despite its limitations, the generalized-cone representation is popular in machine vision.

© 2006 by Taylor & Francis Group, LLC

Skeleton

Skeleton representations use space-curve skeletons. A skeleton can be considered an abstraction of the generalized cone description that consists of only the spines. Skeleton geometry provides useful abstract information. If a radius function is specified at each point on the skeleton, this representation is capable of general-purpose object description.

Multiple Two-Dimensional Projection

For some applications, a library of two-dimensional silhouette projections that represent three-dimensional objects can be conveniently stored. For the recognition of three-dimensional objects with a small number of stable orientations on a flat table, this representation is ideal—if the object silhouettes are sufficiently different. For example, silhouettes have been used to recognize aircraft in any orientation against a well-lit sky background. However, because many different three-dimensional-object shapes can possess the same silhoutte projection, this type of representation is not a general-purpose technique.

Aspect Graphs

In the aspect graph representation, the space of viewpoints is partitioned into maximal regions, where every viewpoint in each region gives the same qualitative view of the object, called the *aspect*. Within each region, **projections** of the object will have the same number and types of features, with identical spatial relationships among them. However, the quantitative properties of these features, such as lengths of edges, vary with the change in viewpoint. Changes in the aspect, called *visual events*, take place at the boundaries between regions. Two aspects are said to be connected by a visual event if their corresponding regions are adjacent in the viewpoint space. An aspect graph is a graph structure whose nodes represent aspects and their associated regions and whose arcs denote the visual events and boundaries between adjacent regions. Figure 19.13(f) shows the aspect graph for a cube.

Characteristic Views

A concept very similar to aspect graphs is that of characteristic views. All of the infinite two-dimensional perspective projection views of an object are grouped into a finite number of topologically equivalent classes. Different views within an equivalence class are related via linear transformations. A representative member of an equivalence class is called the *characteristic view* for that class. In this scheme, objects are assumed to rest on a supporting plane; hence, they are restricted to appear in a number of stable positions. Characteristic views of objects are derived with certain constraints on the camera configuration. It is this use of camera position and orientation information that differentiates the characteristic view representation scheme from the aspect graph. Because characteristic views specify the three-dimensional structure of an object, they also provide a general-purpose object representation.

19.6 Dynamic Vision

Early machine vision systems were concerned primarily with static scenes; however, the world is dynamic. Designing machine vision systems capable of analyzing dynamic scenes is increasingly becoming more popular. For a machine vision system engaged in the performance of nontrivial real-world operations and tasks, the ability to cope with moving and changing objects and viewpoints is vital.

The input to a dynamic-scene analysis system is a sequence of image frames. The camera used to acquire the image sequence may itself be in motion. Each frame represents an image of the scene at a particular instant in time. The changes in a scene may be due to camera motion, object motion, illumination changes, or changes in object structure, size, or shape. Changes in a scene are usually assumed to be due to camera and/or object motion; objects are usually assumed to be either rigid or quasirigid. Other changes are not allowed.

A scene usually contains several objects. An image of the scene at a given time represents a projection of part of the scene; the part of the scene depends on the position of the camera. Four cases represent the

© 2006 by Taylor & Francis Group, LLC

possibilities for the dynamic-camera/world setup:

1. Stationary camera, stationary objects (SCSO)
2. Stationary camera, moving objects (SCMO)
3. Moving camera, stationary objects (MCSO)
4. Moving camera, moving objects (MCMO)

The first case is simple static-scene analysis. In many applications, processing a single image to obtain the required information may be feasible. However, many more applications exist that require information to be extracted from a dynamic environment.

Clearly, a sequence of image frames offers much more information to aid in scene understanding, but significantly increases the amount of data to be processed by the system. Applying static-scene analysis techniques to each frame of a sequence requires an enormous amount of computation, while still suffering from all of the problems of static-scene analysis. Fortunately, research in dynamic-scene analysis has shown that information recovery may be easier in dynamic scenes than in static scenes. In some cases, the total computational effort may be significantly less and the performance is better.

SCMO scenes have received the most attention in dynamic-scene analysis. The objectives of such scene analysis usually are to detect motion, to extract masks of moving objects with the aim of recognizing them, and to compute their motion characteristics. MCMO is the most general case and possibly presents the most difficult situation in dynamic-scene analysis. Many techniques that have been developed assuming a stationary camera are not applicable to a moving camera. Likewise, techniques developed for a moving camera generally assume a stationary scene and are usually not applicable if the objects are moving. SCMO and MCSO have found uses in many applications and have been studied by researchers in various contexts under various assumptions.

Change Detection

Any perceptable motion in a scene results in some change in the sequence of frames of the scene. If such changes are detected, motion characteristics can be analyzed. A good quantitative estimate of the motion components of an object can be obtained if the motion is restricted to a plane that is parallel to the image plane; for three-dimensional motion, only qualitative estimates are possible. By analyzing frame-to-frame differences, a global analysis of the sequence can be performed. Changes can be detected at different levels: pixel, edge, or region.

Difference Pictures

The most obvious method of detecting change between two frames is to directly compare the corresponding pixels of the frames to determine whether or not they are the same. In the simplest form, a binary *difference picture* $DP_{jk}(x, y)$ between frames j and k is obtained by

$$DP_{jk}(x, y) = \begin{cases} 1 & \text{if } |F(x, y, j) - F(x, y, k)| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{19.18}$$

where τ is a threshold and $F(x, y, j)$ and $F(x, y, k)$ are the image arrays of the jth and kth frames, respectively.

In the difference picture, pixels that have a value of 1 may be considered to be the result of object motion. Because of noise, however, such a simple test on real scenes usually produces unsatisfactory results. A simple size filter may be used to ignore pixels not forming a connected cluster of minimal size. Then only those pixels in the difference picture with the value of 1 that belong to a four-connected (or eight-connected) component larger than a set number of pixels will be attributed to object motion. This filter is very effective in reducing noise, but unfortunately it also filters some desirable signals, such as those from small or slowly moving objects.

© 2006 by Taylor & Francis Group, LLC

Likelihood Ratio

To make change detection more robust, regions or groups of pixels at the same location in two frames may be considered and their intensity characteristics may be compared more rigorously. One method using this approach is based on comparing the frames using the *likelihood ratio*. Thus, we can compute the difference picture by replacing $|F(x, y, j) - F(x, y, k)|$ with λ where

$$\lambda = \frac{\left[\frac{\sigma_1^2+\sigma_2^2}{2} + \left(\frac{\mu_1-\mu_2}{2}\right)^2\right]^2}{\sigma_1^2\sigma_2^2} \tag{19.19}$$

and μ and σ denote the mean gray value and the square root of the variance from the frames for the sample area, respectively.

The likelihood ratio can only be applied to regions and not to single pixels. This limitation presents a minor problem, which can be solved by considering corresponding areas of the frames. The likelihood ratio test, combined with a size filter, works well for noise removal in many real-world scenes. The likelihood ratio test can be applied to every point in each frame by considering overlapping areas centered on each pixel of the image, or to a subsampling of points by using nonoverlapping areas, called *superpixels*.

Accumulative Difference Pictures

The problem of missing the detection of small or slowly moving objects can be solved by analyzing change over a series of frames, instead of just between two frames. The *accumulative difference picture* (ADP) is used to detect the motion of small and slowly moving objects. An accumulative difference picture is formed by comparing every frame in an image sequence to a common reference frame. The entry in the accumulative difference picture is increased by one whenever the likelihood ratio for that region exceeds the threshold. Thus, an accumulative difference picture acquired over k frames is given by

$$\begin{aligned} ADP_0(x, y) &= 0 \\ ADP_k(x, y) &= ADP_{k-1}(x, y) + DP_{1k}(x, y) \end{aligned} \tag{19.20}$$

where the first frame of a sequence is usually the reference frame.

Time-Varying Edge Detection

As a result of the importance of edge detection in static scenes, it is reasonable to expect that time-varying edge detection may also be important in dynamic-scene analysis. Moving edges can be detected by combining the spatial and temporal gradients using a logical AND operation, which is implemented through multiplication. Thus, the time-varying edgeness of a point in a frame is given by

$$E_t(x, y, t) = \frac{dF(x, y, t)}{dS}\frac{dF(x, y, t)}{dt} \tag{19.21}$$

where dF/dS and dF/dt are the magnitudes of the spatial and temporal gradients of the intensity at point (x, y, t), respectively. Various conventional edge detectors can be used to compute the spatial gradient, and a simple difference can be used to compute the temporal gradient. In most cases, this edge detector works effectively. By applying a threshold to the product—rather than first differencing and then applying an edge detector, or first detecting edges and then computing their temporal gradient—this method overcomes the problem of missing weak or slowly moving edges.

Optical Flow

Optical flow is the distribution of velocity, relative to the observer, over the points of an image. Optical flow carries information that is valuable in dynamic-scene analysis. Optical flow is determined by the velocity vector at each pixel in an image. Several methods have been devised for calculating optical flow based on two or more frames of a sequence. These methods can be classified into two general categories: *feature*

© 2006 by Taylor & Francis Group, LLC

based and *gradient based*. If a stationary camera is used, most of the points in an image frame will have zero velocity. This is assuming that a very small subset of the scene is in motion, which is usually true. Thus, most applications for optical flow involve a moving camera.

Feature-Based Methods

Feature-based methods for computing optical flow first select some features in consecutive image frames, match these features between frames, and then calculate the disparities between them. As mentioned earlier, the **correspondence problem** can be solved using relaxation. However, the problem of selecting features and establishing correspondence is not easy. Moreover, this method only produces velocity vectors at sparse points within the image.

Gradient-Based Methods

Gradient-based methods exploit the relationship between the spatial and temporal gradients of image intensity. This relationship can be used to segment images based on the velocity of image points. The relationship between the spatial and temporal gradients and the velocity components is

$$F_x u + F_y v + F_t = 0 \tag{19.22}$$

where $u = dx/dt$ and $v = dy/dt$. In this equation, F_x, F_y, and F_t represent the spatial gradients in the x and y directions and the temporal gradient, respectively, and can be computed directly from the image. Then, at every point in an image, there are two unknowns, u and v, yet there is only one equation. Thus, optical flow cannot be directly determined.

The velocity field, however, can be assumed to vary smoothly over an image. Under this assumption, an iterative approach for computing optical flow using two or more frames can be utilized. The following iterative equations are used for the computation of image flow:

$$u = u_{\text{average}} - F_x \frac{P}{D} \qquad v = v_{\text{average}} - F_y \frac{P}{D} \tag{19.23}$$

where

$$P = F_x u_{\text{average}} + F_y v_{\text{average}} + F_t \tag{19.24}$$

$$D = \lambda^2 + F_x^2 + F_y^2 \tag{19.25}$$

where λ is a constant multiplier. When only two frames are used, the computation is iterated over the same frames many times. For more than two frames, each iteration uses a new frame.

Segmentation Using a Moving Camera

If the camera is moving, then every point in the image has nonzero velocity relative to the camera (except the case where object points are moving with the same velocity as the camera). The velocity relative to the camera depends on both the velocity of the point itself and the distance of the point from the camera. Approaches based on differences may be extended for segmenting moving-camera scenes. If the aim is to extract images of moving objects, however, then additional information is required to decide whether the motion at a point is due solely to its depth or is due to a combination of its depth and its motion. Gradient-based approaches will also require additional information.

If the camera's direction of motion is known, the *focus of expansion* (FOE) with respect to the stationary components in the scene can easily be computed. The FOE will have coordinates (x_f, y_f) where

$$x_f = \frac{dx}{dz} \qquad y_f = \frac{dy}{dz} \tag{19.26}$$

in the image plane. The velocity vectors of all stationary points in a scene project onto the image plane so that they intersect at the FOE. A transformation with respect to the FOE can be used to simplify the task

© 2006 by Taylor & Francis Group, LLC

of segmentation. The *ego-motion polar* (EMP) transformation of an image transforms a frame $F(x, y, t)$ into $E(r, \theta, t)$ using

$$E(r, \theta, t) = F(x, y, t) \tag{19.27}$$

where

$$r = \sqrt{(x - x_f)^2 + (y - y_f)^2} \tag{19.28}$$

and

$$\theta = \tan^{-1}\left[\frac{(y - y_f)}{(x - x_f)}\right] \tag{19.29}$$

In EMP space, stationary points are displaced only along the θ axis between the frames of an image sequence, whereas points on moving objects are displaced along the r axis as well as the θ axis. Thus, the displacement in EMP space can be used to segment a scene into its stationary and nonstationary components. Moreover, when *complex logarithmic mapping* (CLM) is performed about the FOE, interesting points in the EMP space can be obtained.

19.7 Applications

Machine vision systems have uses in a wide variety of disciplines, from medicine to robotics, from automatic inspection to autonomous navigation, and from document analysis to multimedia systems. New machine vision systems are constantly emerging and becoming a part of everyday life. In this section, we present a brief description some of the various applications of machine vision.

Optical Character Recognition (OCR) and Document Image Analysis

The objective of document image analysis is to recognize the text and graphics components in images and extract the intended information as a human would. Two categories of document processing can be defined, *textual processing* and *graphics processing*. Textual processing deals with the text components of a document image. Some tasks here are recognizing the text by optical character recognition (OCR); skew detection and correction (any tilt at which the documents may have been scanned); and finding the columns, paragraphs, text lines, and words. Graphics processing deals with the nontextual components of a document image, such as lines and symbols that make up line diagrams, delimiting lines between text sections, and company logos.

Document analysis is currently a very active area of research. Researchers have devised systems ranging from automatic engineering drawing interpretation and recognition of tabular drawings to the recognition of zip codes from postal envelopes and the interpretation of musical scores. However, the real success story of document analysis is OCR. This is the one area of machine vision in which scientific study has lead to numerous low-cost marketed products. Many of the current OCR systems report accuracies well in the upper 90th percentile.

Many document analysis publications can be found in the journals and conference proceedings listed in the Further Information section. In addition, the International Conference on Document Analysis and Recognition (ICDAR) and the International Workshop or Graphics Recognition (IWGR) are biannual meetings held in conjunction with each other dedicated entirely to the field of document analysis.

Medical Image Analysis

Medical imaging analysis deals primarily with images such as X-rays, computerized tomograph (CT) scans, and magnetic resonance imaging (MRI) images. Early work in medical image analysis overlapped with that of image processing; the main task was to enhance medical images for viewing by a physician; no automatic interpretation or high-level reasoning by the system was involved. However, more recent work is being

© 2006 by Taylor & Francis Group, LLC

conducted on medical imagery, which more closely fits the definition of machine vision; for example, systems to search images for diseased organs or tissues based on known models (images) and features of the diseased samples and systems to generate three-dimensional models of organs based on CT scan and MRI. Some active areas of research include the use of three-dimensional images in surgery and surgical planning (especially neurosurgery), virtual surgery, and generating time-varying three-dimensional models obtained from sequences of images (e.g., pumping heart). These systems encompass many of the aspects of typical machine vision systems plus incorporate many aspects of computer graphics.

In addition to the listings at the end of this section, the following are excellent sources of information on current research in medical imaging: *IEEE Transactions on Medical Imaging*, *IEEE Transactions on Biomedical Engineering*, *IEEE Transactions on Image Processing*, and *IEEE Engineering in Medicine and Biology Magazine*. There are also a number of conferences dedicated to medical imaging research: SPIE Medical Imaging, IEEE Engineering in Medicine and Biology, Medicine Meets Virtual Reality, select sessions/symposium of IEEE Visualization, and SPIE Visualization in Biomedical Computing.

Photogrammetry and Aerial Image Analysis

Photogrammetry deals with the task of making reliable measurements from images. In the early days of photogrammetry, the images were actual printed photographs often taken from balloons. Today, however, the remote sensing process is multispectral using energy in many other parts of the electromagnetic spectrum, such as ultraviolet and infrared. The images are often transmitted directly from satellites orbiting the globe, such as the Landsat satellites first launched in the early 1970s. Some of the applications of photogrammetry and aerial image analysis include atmospheric exploration; thermal image analysis for energy conservation; monitoring natural resources, crop conditions, land cover, and land use; weather prediction; pollution studies; urban planning; military reconnaissance; plus many others in geology, hydrology, and oceanography.

There are several organizations dedicated to the study of these types of remote sensing tasks. The following are very good sources for information on photogrammetry and aerial image analysis: The American Society of Photogrammetry, The International Remote Sensing Institute (ISRI), and The American Congress on Surveying and Mapping. In addition, there are several conferences and symposia dealing with this topic: SPIE Remote Sensing, International Geosciences and Remote Sensing Symposium, *IEEE Transactions on Geoscience and Remote Sensing*, and the Symposium on Remote Sensing sponsored by the American Geological Institute.

Industrial Inspection and Robotics

Unlike many of the already mentioned machine vision tasks, automatic inspection and robotics perform many tasks in real time that significantly increases the complexity of the system. The general goal of such systems is sensor-guided control. Typical industrial applications include automatic inspection of machined parts, solder joint surfaces (welds), silicon wafers, produce, and even candy. Some of the challenges faced by industrial vision system designers include determining the optimal configuration of the camera and lighting, determining the most suitable color space representation of the illumination, modeling various surface reflectance mechanisms, dynamic sensor feedback, real-time manipulator control, real-time operating system interfaces, and neural networks.

Structured lighting techniques have been used extensively in industrial vision applications where the illumination of the scene can be easily controlled. In a typical application, objects on a conveyor belt pass through a plane of light, creating a distortion in the image of the light stripe. The profile of the object at the plane of the light beam is then calculated. This process is repeated at regular intervals as the object moves along the conveyor belt to recover the shape of the object. Then appropriate actions are taken depending on the goal of the system.

The following are sources dedicated primarily to industrial vision applications and robotics: *International Journal of Robotics Research*, *IEEE Transactions on Robotics and Automation*, IEEE's International Conference on Robotics and Automation, and *IEEE Transactions on Systems, Man, and Cybernetics*.

© 2006 by Taylor & Francis Group, LLC

Autonomous Navigation

Closely related to robotics is the area of autonomous navigation. Much work is being done to develop systems to enable robots or other mobile vehicles to automatically navigate through a specific environment. Techniques involved include active vision (sensor control), neural networks, high-speed stereo vision, three-dimensional vision (range imaging), high-level reasoning for navigational planning, and signal compression and transmission for accurate remote vehicle control.

Visual Information Management Systems

Probably one of the newest areas of machine vision research is in visual information management systems (VIMS). With the recent advances in low-cost computing power and the ever increasing number of multimedia applications, digital imagery is becoming a larger and larger part of every day life. Research in VIMS is providing methods to handle all of this digital information. Such VIMS applications include interactive television, video teleconferencing, digital libraries, video-on-demand, and large-scale video databases. Image processing and machine vision play a very important role in much of this work ranging anywhere from designing video compression schemes, which allow many processing techniques to be performed directly on the compressed datastream and developing more efficient indexing methods for multidimensional data, to automatic scene cut detection for automatically indexing large stockpiles of video data and developing methods to query image databases by image content as opposed to standard structured grey language (SQL) techniques.

Defining Terms

Correspondence problem: The problem of matching points in one image with their corresponding points in a second image.

Histogram: A plot of the frequency of occurrence of the grey levels in an image.

Quantization: The process of representing the continuous range of image intensities by a limited number of discrete grey values.

Sampling: The process of representing the continuous image intensity function as a discrete two-dimensional array.

Segmentation: The process of separating the objects from the background.

Polygonalization: A method of representing a contour by a set of connected straight line segments; for closed curves, these segments form a polygon.

Projection: The transformation and representation of a high-dimensional space in a lesser number of dimensions (i.e., a three-dimensional scene represented as a two-dimensional image).

Thresholding: A method of separating objects from the background by selecting an interval (usually in pixel intensity) and setting any points within the interval to 1 and points outside the interval to 0.

References

Besl, P.J. 1988. Active, Optical range imaging sensors. *Machine Vision and Applications* 1(2):127–152.

Besl, P. and Jain, R.C. 1985. Three dimensional object recognition. *ACM Computing Surveys* 17(1).

Duda, R.O. and Hart, P.E. 1973. *Pattern Classification and Scene Analysis*. Wiley, New York.

Foley, J.D., van Dam, A., Feiner, S.K., and Hughes, J.F. 1990. *Computer Graphics, Principles and Practice*. Addison-Wesley, Reading, MA.

Gonzalez, R.C. and Woods, R.E. 1992. *Digital Image Processing*. Addison-Wesley, Reading, MA.

Javis, R.A. 1983. A perspective on range finding techniques for computer vision. *TEEE Trans. on Pattern Analysis and Machine Intelligence* 5(2):122–139.

Kasturi, R. and Jain, R.C. 1991. *Computer Vision: Principles*. IEEE Computer Society Press.

Kosko, B. 1992. *Neural Networks and Fuzzy Systems*. Prentice-Hall, Englewood Cliffs, NJ.

© 2006 by Taylor & Francis Group, LLC

Jain, R., Kasturi, R., and Schunck, B.G. 1995. *Machine Vision*. McGraw-Hill, New York.
Marr, D. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, San Francisco, CA.
Tanimoto, S.L. 1995. *The Elements of Artificial Intelligence Using Common Lisp*. Computer Science Press.
Winston, P.H. 1992. *Artificial Intelligence*. Addison-Wesley, Reading, MA.

Further Information

Much of the material presented in this section has been adapted from:

Jain, R., Kasturi, R., and Schunck, B.G. 1995. *Machine Vision*. McGraw-Hill, New York.
Kasturi, R. and Jain, R.C. 1991. *Computer Vision: Principles*. IEEE Computer Society Press, Washington.

In addition the following books are reccommended:

Rosenfeld, A. and Kak, A.C. 1982. *Digital Picture Processing*. Academic Press, Englewood Cliffs, NJ.
Jain, A.K. 1989. *Fundamentals of Digital Image Processing*. Prentice-Hall, New York.
Haralick, R.M. and Shapiro, L.G. 1992–1993. *Computer and Robot Vision*. Addison-Wesley, Reading, MA.
Horn, B. 1986. *Robot Vision*. McGraw-Hill.
Schalkoff, R.J. 1989. *Digital Image Processing and Computer Vision*. Wiley, New York.

Additional information on machine vision can be found in the following technical journals:

Artificial Intelligence (*AI*)
Computer Vision, Graphics, and Image Processing (*CVGIP*)
IEEE Transactions on Image Processing
IEEE Transactions on Pattern Analysis and Machine Intelligence (*PAMI*)
Image and Vision Computing
International Journal of Computer Vision
International Journal on Pattern Recognition and Artificial Intelligence
Machine Vision and Applications (*MVA*)
Pattern Recognition (*PR*)
Pattern Recognition Letters (*PRL*)

The following conference proceedings are also a good source for machine vision information:

A series of conferences by the International Society for Optical Engineering (SPIE)
A series of workshops by the Institute of Electrical and Electronic Engineering (IEEE) Computer Society
A series of workshops by the International Association for Pattern Recognition (IAPR)
European Conference on Computer Vision (ECCV)
IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
International Conference on Computer Vision (ICCV)
International Conference on Pattern Recognition (ICPR)

© 2006 by Taylor & Francis Group, LLC

20

A Brief Survey of Speech Enhancement[1]

Yariv Ephraim
Hanoch Lev-Ari
William J.J. Roberts

20.1 Introduction **20**-1
20.2 The Signal Subspace Approach **20**-2
20.3 Short-Term Spectral Estimation **20**-5
20.4 Multiple State Speech Model **20**-6
20.5 Second-Order Statistics Estimation **20**-7
20.6 Concluding Remarks **20**-9

20.1 Introduction

Speech enhancement aims at improving the performance of speech communication systems in noisy environments. Speech enhancement may be applied, for example, to a mobile radio communication system, a speech recognition system, a set of low quality recordings, or to improve the performance of aids for the hearing impaired. The interference source may be a wide-band noise in the form of a white or colored noise, a periodic signal such as in hum noise, room reverberations, or it can take the form of fading noise. The first two examples represent additive noise sources, while the other two examples represent convolutional and multiplicative noise sources, respectively. The speech signal may be simultaneously attacked by more than one noise source.

There are two principal perceptual criteria for measuring the performance of a speech enhancement system. The *quality* of the enhanced signal measures its clarity, distorted nature, and the level of residual noise in that signal. The quality is a *subjective* measure that is indicative of the extent to which the listener is comfortable with the enhanced signal. The second criterion measures the *intelligibility* of the enhanced signal. This is an *objective* measure which provides the percentage of words that could be correctly identified by listeners. The words in this test need not be meaningful. The two performance measures are not correlated. A signal may be of good quality and poor intelligibility and vice versa. Most speech enhancement systems improve the quality of the signal at the expense of reducing its intelligibility. Listeners can usually extract more information from the noisy signal than from the enhanced signal by careful listening to that signal. This is obvious from the *data processing theorem* of information theory. Listeners, however, experience fatigue over extended listening sessions, a fact that results in reduced intelligibility of the noisy signal. Is such situations, the intelligibility of the enhanced signal may be higher than that of the noisy signal. Less effort would usually be required from the listener to decode portions of the enhanced signal that correspond to high signal to noise ratio segments of the noisy signal.

[1] This paper is partially based on "Extension of the Signal Subspace Speech Enhancement Approach to Colored Noise" by Hanoch Lev-Ari and Yariv Ephraim which appeared in *IEEE Sig. Proc. Let.*, vol. 10, pp. 104–106, April 2003 ©2003 IEEE.

© 2006 by Taylor & Francis Group, LLC

Both the quality and intelligibility are elaborate and expensive to measure, since they require listening sessions with live subjects. Thus, researchers often resort to less formal listening tests to assess the quality of an enhanced signal, and they use automatic speech recognition tests to assess the intelligibility of that signal. Quality and intelligibility are also hard to quantify and express in a closed form that is amenable to mathematical optimization. Thus, the design of speech enhancement systems is often based on mathematical measures that are somehow believed to be correlated with the quality and intelligibility of the speech signal. A popular example involves estimation of the clean signal by minimizing the mean square error (MSE) between the logarithms of the spectra of the original and estimated signals [5]. This criterion is believed to be more perceptually meaningful than the minimization of the MSE between the original and estimated signal waveforms [13].

Another difficulty in designing efficient speech enhancement systems is the lack of *explicit* statistical models for the speech signal and noise process. In addition, the speech signal, and possibly also the noise process, are not strictly stationary processes. Common parametric models for speech signals, such as an autoregressive process for short-term modeling of the signal, and a hidden Markov process (HMP) for long-term modeling of the signal, have not provided adequate models for speech enhancement applications. A variant of the *expectation-maximization* (EM) algorithm, for maximum likelihood (ML) estimation of the autoregressive parameter from a noisy signal, was developed by Lim and Oppenheim [12] and tested in speech enhancement. Several estimation schemes, which are based on hidden Markov modeling of the clean speech signal and of the noise process, were developed over the years, see, e.g., Ephraim [6]. In each case, the HMPs for the speech signal and noise process were designed from training sequences from the two processes, respectively. While autoregressive and hidden Markov models have proved extremely useful in coding and recognition of *clean* speech signals, respectively, they were not found to be sufficiently refined models for speech enhancement applications.

In this chapter we review some common approaches to speech enhancement that were developed primarily for additive wide-band noise sources. Although some of these approaches have been applied to reduction of reverberation noise, we believe that the dereverberation problem requires a completely different approach that is beyond the scope of this capter. Our primary focus is on the spectral subtraction approach [13] and some of its derivatives such as the signal subspace approach [7] [11], and the estimation of the short-term spectral magnitude [16, 4, 5]. This choice is motivated by the fact that some derivatives of the spectral subtraction approach are still the best approaches available to date. These approaches are relatively simple to implement and they usually outperform more elaborate approaches which rely on parametric statistical models and training procedures.

20.2 The Signal Subspace Approach

In this section we present the principles of the signal subspace approach and its relations to Wiener filtering and spectral subtraction. Our presentation follows [7] and [11]. This approach assumes that the signal and noise are noncorrelated, and that their second-order statistics are available. It makes no assumptions about the distributions of the two processes.

Let Y and W be k-dimensional random vectors in a Euclidean space $\mathcal{R}^k$ representing the clean signal and noise, respectively. Assume that the expected value of each random variable is zero in an appropriately defined probability space. Let $Z = Y + W$ denote the noisy vector. Let R_y and R_w denote the covariance matrices of the clean signal and noise process, respectively. Assume that R_w is positive definite. Let H denote a $k \times k$ real matrix in the linear space $\mathcal{R}^{k\times k}$, and let $\hat{Y} = HZ$ denote the linear estimator of Y given Z. The residual signal in this estimation is given by

$$Y - \hat{Y} = (I - H)Y - HW \tag{20.1}$$

where I denotes, as usual, the identity matrix. To simplify notation, we shall not explicitly indicate the dimensions of the identity matrix. These dimensions should be clear from the context. In Eq. (20.1),

© 2006 by Taylor & Francis Group, LLC

$D = (I - H)Y$ is the signal distortion and $N = HW$ is the residual noise in the linear estimation. Let $(\cdot)'$ denote the transpose of a real matrix or the conjugate transpose of a complex matrix. Let

$$\overline{\epsilon_d^2} = \frac{1}{k}\text{tr}E\{DD'\} = \frac{1}{k}\text{tr}\{(I - H)R_y(I - H)'\} \tag{20.2}$$

denote the average signal distortion power where $\text{tr}\{\cdot\}$ denotes the trace of a matrix. Similarly, let

$$\overline{\epsilon_n^2} = \frac{1}{k}\text{tr}E\{NN'\} = \frac{1}{k}\text{tr}\{HR_wH'\} \tag{20.3}$$

denote the average residual noise power.

The matrix H is estimated by minimizing the signal distortion $\overline{\epsilon_d^2}$ subject to a threshold on the residual noise power $\overline{\epsilon_n^2}$. It is obtained from

$$\begin{gathered}\min_H \overline{\epsilon_d^2} \\ \text{subject to : } \overline{\epsilon_n^2} \leq \alpha\end{gathered} \tag{20.4}$$

for some given α. Let $\mu \geq 0$ denote the Lagrange multiplier of the inequality constraint. The optimal matrix, say $H = H_1$, is given by

$$H_1 = R_y(R_y + \mu R_w)^{-1} \tag{20.5}$$

The matrix H_1 can be implemented as follows. Let $R_w^{1/2}$ denote the symmetric positive definite square root of R_w and let $R_w^{-1/2} = (R_w^{1/2})^{-1}$. Let U denote an orthogonal matrix of eigenvectors of the symmetric matrix $R_w^{-1/2}R_yR_w^{-1/2}$. Let $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_k)$ denote the diagonal matrix of nonnegative eigenvalues of $R_w^{-1/2}R_yR_w^{-1/2}$. Then

$$H_1 = R_w^{1/2}U\Lambda(\Lambda + \mu I)^{-1}U'R_w^{-1/2} \tag{20.6}$$

When H_1 in Eq. (20.6) is applied to Z, it first whitens the input noise by applying $R_w^{-1/2}$ to Z. Then, the orthogonal transformation U' corresponding to the covariance matrix of the whitened clean signal is applied, and the transformed signal is modified by a diagonal Wiener-type gain matrix.

In Eq. (20.6), components of the whitened noisy signal that contain noise only are nulled. The indices of these components are given by $\{j : \lambda_j = 0\}$. When the noise is white, $R_w = \sigma_w^2 I$, and U and Λ are the matrices of eigenvectors and eigenvalues of R_y/σ_w^2, respectively. The existence of null components $\{j : \lambda_j = 0\}$ for the signal means that the signal lies in a subspace of the Euclidean space $\mathcal{R}^k$. At the same time, the eigenvalues of the noise are all equal to σ_w^2 and the noise occupies the entire space $\mathcal{R}^k$. Thus, the signal subspace approach first eliminates the noise components outside the signal subspace and then modifies the signal components inside the signal subspace in accordance with the criterion of Eq. (20.4).

When the signal and noise are *wide-sense stationary*, the matrices R_y and R_w are Toeplitz with associated power spectral densities of $f_y(\theta)$ and $f_w(\theta)$, respectively. The angular frequency θ lies in $[0, 2\pi)$. When the signal and noise are *asymptotically weakly stationary*, the matrices R_y and R_w are asymptotically Toeplitz and have the associated power spectral densities $f_y(\theta)$ and $f_w(\theta)$, respectively [10]. Since the latter represents a somewhat more general situation, we proceed with asymptotically weakly stationary signal and noise. The filter H_1 in Eq. (20.5) is then asymptotically Toeplitz with associated power spectral density

$$h_1(\theta) = \frac{f_y(\theta)}{f_y(\theta) + \mu f_w(\theta)} \tag{20.7}$$

This is the noncausal Wiener filter for the clean signal with an adjustable noise level determined by the constraint α in Eq. (20.4). This filter is commonly implemented using estimates of the two power spectral densities. Let $\hat{f}_y(\theta)$ and $\hat{f}_w(\theta)$ denote the estimates of $f_y(\theta)$ and $f_w(\theta)$, respectively. These estimates could, for example, be obtained from the periodogram or the smoothed periodogram. In that case, the filter is

© 2006 by Taylor & Francis Group, LLC

implemented as

$$\hat{h}_1(\theta) = \frac{\hat{f}_y(\theta)}{\hat{f}_y(\theta) + \mu \hat{f}_w(\theta)} \tag{20.8}$$

When $\hat{f}_y(\theta)$ is implemented as

$$\hat{f}_y(\theta) = \begin{cases} \hat{f}_z(\theta) - \hat{f}_w(\theta) & \text{if nonnegative} \\ \varepsilon & \text{otherwise} \end{cases} \tag{20.9}$$

then a *spectral subtraction* estimator for the clean signal results. The constant $\varepsilon \geq 0$ is often referred to as a "spectral floor." Usually $\mu \geq 2$ is chosen for this estimator.

The enhancement filter H could also be designed by imposing constraints on the spectrum of the residual noise. This approach enables shaping of the spectrum of the residual noise to minimize its perceptual effect. Suppose that a set $\{v_i, i = 1, \ldots, m\}$, $m \leq k$, of k-dimensional real or complex orthonormal vectors, and a set $\{\alpha_i, i = 1, \ldots, m\}$ of non-negative constants, are chosen. The vectors $\{v_i\}$ are used to transform the residual noise into the spectral domain, and the constants $\{\alpha_i\}$ are used as upper bounds on the variances of these spectral components. The matrix H is obtained from

$$\begin{gathered} \min_H \overline{\epsilon_d^2} \\ \text{subject to : } E\{|v_i' N|^2\} \leq \alpha_i, \quad i = 1, \ldots, m \end{gathered} \tag{20.10}$$

When the noise is white, the set $\{v_i\}$ could be the set of eigenvectors of R_y and the variances of the residual noise along these coordinate vectors are constrained. Alternatively, the set $\{v_i\}$ could be the set of orthonormal vectors related to the DFT. These vectors are given by $v_i' = k^{-1/2}(1, e^{-j\frac{2\pi}{k}(i-1)\cdot 1}, \ldots, e^{-j\frac{2\pi}{k}(i-1)\cdot(k-1)})$. Here we must choose $\alpha_i = \alpha_{k-i+2}, i = 2, \ldots, k/2$, assuming k is even, for the residual noise power spectrum to be symmetric. This implies that at most $k/2 + 1$ constraints can be imposed. The DFT-related $\{v_i\}$ enable the use of constraints that are consistent with auditory perception of the residual noise.

To present the optimal filter, let e_l denote a unit vector in $\mathcal{R}^k$ for which the lth component is one and all other components are zero. Extend $\{v_1, \ldots, v_m\}$ to a $k \times k$ orthogonal or unitary matrix $V = (v_1, \ldots, v_k)$. Set $\mu_i = 0$ for $m < i \leq k$, and let $M = \text{diag}(k\mu_1, \ldots, k\mu_k)$ denote the matrix of k times the Lagrange multipliers which are assumed nonnegative. Define $Q = R_w^{-1/2} U$ and $T = Q'V$. The optimal estimation matrix, say $H = H_2$, is given by [11]

$$H_2 = R_w^{1/2} U \tilde{H}_2 U' R_w^{-1/2} \tag{20.11}$$

where the columns of $\tilde{H}_2$ are given by

$$\tilde{h}_l = T\lambda_l (M + \lambda_l I)^{-1} T^{-1} e_l, \quad l = 1, \ldots, k \tag{20.12}$$

provided that $k\mu_i \neq -\lambda_l$ for all $i.l$. The optimal estimator first whitens the noise, then applies the orthogonal transformation U' obtained from eigendecomposition of the covariance matrix of the whitened signal, and then modifies the resulting components using the matrix $\tilde{H}_2$. This is analogous to the operation of the estimator H_1 in Eq. (20.6). The matrix $\tilde{H}_2$, however, is not diagonal when the input noise is colored.

When the noise is white with variance σ_w^2 and $V = U$ and $m = k$ are chosen, the optimization problem of Eq. (20.10) becomes trivial since knowledge of input and output noise variances uniquely determines the filter H. This filter is given by $H = UGU'$ where $G = \text{diag}(\sqrt{\alpha_1}, \ldots, \sqrt{\alpha_k})$ [7]. For this case, the heuristic choice of

$$\alpha_i = \exp\{-\nu \sigma_w^2 / \lambda_i\} \tag{20.13}$$

where $\nu \geq 1$ is an experimental constant and was found useful in practice [7]. This choice is motivated by the observation that for $\nu = 2$, the first order Taylor expansion of $\alpha_i^{-1/2}$ leads to an estimator $H = UGU'$

© 2006 by Taylor & Francis Group, LLC

which coincides with the Wiener estimation matrix in Eq. (20.6) with $\mu = 1$. The estimation matrix using Eq. (20.13) performs significantly better than the Wiener filter in practice.

20.3 Short-Term Spectral Estimation

In another earlier approach for speech enhancement, the short-time spectral magnitude of the clean signal is estimated from the noisy signal. The speech signal and noise process are assumed statistically independent, and spectral components of each of these two processes are assumed zero mean statistically independent Gaussian random variables. Let $A_y e^{j\theta_y}$ denote a spectral component of the clean signal Y in a given frame. Let $A_z e^{j\theta_z}$ denote the corresponding spectral component of the noisy signal. Let $\sigma_y^2 = E\{A_y^2\}$ and $\sigma_z^2 = E\{A_z^2\}$ denote, respectively, the variances of the clean and noisy spectral components. If the variance of the corresponding spectral component of the noise process in that frame is denoted by σ_w^2, then we have $\sigma_z^2 = \sigma_y^2 + \sigma_w^2$. Let

$$\xi = \frac{\sigma_y^2}{\sigma_w^2}; \quad \gamma = \frac{A_z^2}{\sigma_w^2}; \quad \vartheta = \frac{\xi}{\xi + 1}\gamma \tag{20.14}$$

The MMSE estimation of A_y from $A_z e^{j\theta_z}$ is given by [4]

$$\hat{A}_y = \Gamma(1.5)\frac{\sqrt{\vartheta}}{\gamma}\exp\left(-\frac{\vartheta}{2}\right)\left[(1+\vartheta)I_0\left(\frac{\vartheta}{2}\right) + \vartheta I_1\left(\frac{\vartheta}{2}\right)\right]A_z \tag{20.15}$$

where $\Gamma(1.5) = \frac{\sqrt{\pi}}{2}$, and $I_0(\cdot)$ and $I_1(\cdot)$ denote, respectively, the modified Bessel functions of the zeroth and first order. Similarly to the Wiener filter given in Eq. (20.5), this estimator requires knowledge of second order statistics of each signal and noise spectral components, σ_y^2 and σ_w^2, respectively.

To form an estimator for the spectral component of the clean signal, the spectral magnitude estimator Eq. (20.15) is combined with an estimator of the complex exponential of that component. Let $\widehat{e^{j\theta_y}}$ be an estimator of $e^{j\theta_y}$. This estimator is a function of the noisy spectral component $A_z e^{j\theta_z}$. MMSE estimation of the complex exponential $e^{j\theta_y}$ is obtained from

$$\begin{gathered}\min_{\widehat{e^{j\theta_y}}} E\{|e^{j\theta_y} - \widehat{e^{j\theta_y}}|^2\} \\ \text{subject to : } |\widehat{e^{j\theta_y}}| = 1\end{gathered} \tag{20.16}$$

The constraint in Eq. (20.16) ensures that the estimator $\widehat{e^{j\theta_y}}$ does not affect the optimality of the estimator $\hat{A}_y$ when the two are combined. The constrained minimization problem in Eq. (20.16) results in the estimator $\widehat{e^{j\theta_y}} = e^{j\theta_z}$ which is simply the complex exponential of the noisy signal.

Note that the Wiener filter Eq. (20.8) has zero phase and hence it effectively uses the complex exponential of the noisy signal $e^{j\theta_z}$ in estimating the clean signal spectral component. Thus, both the Wiener estimator of Eq. (20.8) and the MMSE spectral magnitude estimator of Eq. (20.15) use the complex exponential of the noisy phase. The spectral magnitude estimate of the clean signal obtained by the Wiener filter, however, is not optimal in the MMSE sense.

Other criteria for estimating A_y could also be used. For example, A_y could be estimated from

$$\min_{\hat{A}_y} E\{(\log A_y - \log \hat{A}_y)^2\} \tag{20.17}$$

This criterion aims at producing an estimate of A_y whose logarithm is as close as possible to the logarithm of A_y in the MMSE sense. This perceptually motivated criterion results in the estimator given by [5]

$$\hat{A}_y = \frac{\sigma_y^2}{\sigma_y^2 + \sigma_w^2}\exp\left(\frac{1}{2}\int_{\vartheta}^{\infty}\frac{e^{-t}}{t}dt\right)A_z \tag{20.18}$$

The integral in Eq. (20.18) is the well known exponential integral of ϑ and it can be numerically evaluated.

© 2006 by Taylor & Francis Group, LLC

In another example, A_y^2 could be estimated from

$$\min_{\hat{A}_y} E\{(A_y^2 - \widehat{A_y^2})^2\} \tag{20.19}$$

and an estimate of A_y could be obtained from $\sqrt{\widehat{A_y^2}}$. The criterion in Eq. (20.19) aims at estimating the magnitude squared of the spectral component of the clean signal in the MMSE sense. This estimator is particularly useful when subsequent processing of the enhanced signal is performed, for example, in autoregressive analysis for low bit rate signal coding applications [13]. In that case, an estimator of the autocorrelation function of the clean signal can be obtained from the estimator $\widehat{A_y^2}$. The optimal estimator in the sense of Eq. (20.19) is well known and is given by (see e.g., [6])

$$\widehat{A_y^2} = \frac{\sigma_y^2 \sigma_w^2}{\sigma_y^2 + \sigma_w^2} + \left| \frac{\sigma_y^2}{\sigma_y^2 + \sigma_w^2} A_z \right|^2 \tag{20.20}$$

20.4 Multiple State Speech Model

All estimators presented in Section 20.2 and Section 20.3 make the implicit assumption that the speech signal is always present in the noisy signal. In the notation of Section 20.2, $Z = Y + W$. Since the length of a frame is relatively short, in the order of $30 - 50$ msec, it is more realistic to assume that speech may be present in the noisy signal with some probability, say η, and may be absent from the noisy signal with one minus that probability. Thus we have two hypotheses, one of speech presence, say H_1, and the other of speech absence, say H_0, that occur with probabilities η and $1 - \eta$, respectively. We have

$$Z = \begin{cases} Y + W & \text{under } H_1 \\ W & \text{under } H_0 \end{cases} \tag{20.21}$$

The MMSE estimator of A_y under the uncertainty model can be shown to be

$$\begin{aligned} \tilde{A}_y &= \Pr(H_1|Z) \cdot E\{A_y|Z, H_1\} + \Pr(H_0|Z) \cdot E\{A_y|Z, H_0\} \\ &= \Pr(H_1|Z) \cdot \hat{A}_y \end{aligned} \tag{20.22}$$

since $E\{A_y|Z, H_0\} = 0$ and $E\{A_y|Z, H_1\} = \hat{A}_y$ as given by Eq. (20.15). The more realistic estimator given in Eq. (20.22) was found useful in practice as it improved the performance of the estimator in Eq. (20.15). Other estimators may be derived under this model. Note that the model as in Eq. (20.21) is not applicable to the estimator in Eq. (20.18) since A_y must be positive for the criterion in Eq. (20.17) to be meaningful. Speech enhancement under the speech presence uncertainty model was first proposed and applied by McAulay and Malpass in their pioneering work [16].

An extension of this model leads to the assumption that speech vectors may be in different states at different time instants. The speech presence uncertainty model assumes two states representing speech presence and speech absence. In another model of Drucker [3], five states were proposed representing fricative, stop, vowel, glide, and nasal speech sounds. A speech absence state could be added to that model as a sixth state. This model requires an estimator for each state just as in Eq. (20.22).

A further extension of these ideas, in which multiple states that evolve in time are possible, is obtained when one models the speech signal by a hidden Markov process (HMP) [8]. An HMP is a bivariate random process of states and observations sequences. The state process $\{S_t, t = 1, 2, \ldots\}$ is a finite-state homogeneous Markov chain that is not directly observed. The observation process $\{Y_t, t = 1, 2, \ldots\}$ is conditionally independent given the state process. Thus, each observation depends statistically only on the state of the Markov chain at the same time and not on any other states or observations. Consider, for example, an HMP observed in an additive white noise process $\{W_t, t = 1, 2, \ldots\}$. For each t, let

© 2006 by Taylor & Francis Group, LLC

$Z_t = Y_t + W_t$ denote the noisy signal. Let $Z^t = \{Z_1, \ldots, Z_t\}$. Let J denote the number of states of the Markov chain. The causal MMSE estimator of Y_t given $\{Z^t\}$ is given by [6]

$$\begin{aligned}\hat{Y}_t &= E\{Y_t|Z^t\} \\ &= \sum_{j=1}^{J} \Pr(S_t = j|Z^t) E\{Y_t|S_t = j, Z_t\}\end{aligned} \tag{20.23}$$

The estimator as given in Eq. (20.23) reduces to Eq. (20.22) when $J = 2$ and $\Pr(S_t = j|Z^t) = \Pr(S_t = j|Z_t)$, or when the states are statistically independent of each other.

An HMP is a parametric process that depends on the initial distribution and transition matrix of the Markov chain and on the parameter of the conditional distributions of observations given states. The parameter of an HMP can be estimated off-line from training data and then used in constructing the estimator in Eq. (20.23). This approach has a great theoretical appeal since it provides a solid statistical model for speech signals. It also enjoys a great intuitive appeal since speech signals do cluster into sound groups of distinct nature, and dedication of a filter for each group is appropriate. The difficulty in implementing this approach is in achieving low probability of error in mapping vectors of the noisy speech onto states of the HMP. Decoding errors result in wrong association of speech vectors with the set of predesigned estimators $\{E\{Y_t|S_t = j, Z_t\}, j = 1, \ldots, J\}$, and thus in poorly filtered speech vectors. In addition, the complexity of the approach grows with the number of states, since each vector of the signal must be processed by all J filters.

The approach outlined above could be applied based on other models for the speech signal. For example, in [20], a harmonic process based on an estimated pitch period was used to model the speech signal.

20.5 Second-Order Statistics Estimation

Each of the estimators presented in Section 20.2 and Section 20.3 depends on some statistics of the clean signal and noise process which are assumed known *a-priori*. The signal subspace estimators as in Eq. (20.5) and Eq. (20.11) require knowledge of the covariance matrices of the signal and noise. The spectral magnitude estimators in Eq. (20.15), Eq. (20.18), and Eq. (20.20) require knowledge of the variance of each spectral component of the speech signal and of the noise process. In the absence of explicit knowledge of the second-order statistics of the clean signal and noise process, these statistics must be estimated either from training sequences or directly from the noisy signal.

We note that when estimates of the second-order statistics of the signal and noise replace the true second-order statistics in a given estimation scheme, the optimality of that scheme can no longer be guaranteed. The quality of the estimates of these second-order statistics is key for the overall performance of the speech enhancement system. Estimation of the second-order statistics can be performed in various ways usually outlined in the theory of spectral estimation [19]. Some of these approaches are reviewed in this section.

Estimation of speech signals second-order statistics from training data has proven successful in coding and recognition of clean speech signals. This is commonly done in coding applications using vector quantization and in recognition applications using hidden Markov modeling. When only noisy signals are available, however, matching of a given speech frame to a codeword of a vector quantizer or to a state of an HMP is susceptible to decoding errors. The significance of these errors is the creation of a mismatch between the true and estimated second order statistics of the signal. This mismatch results in application of wrong filters to frames of the noisy speech signal, and in unacceptable quality of the processed noisy signal.

On-line estimation of the second-order statistics of a speech signal from a sample function of the noisy signal has proven to be a better choice. Since the analysis frame length is usually relatively small, the covariance matrices of the speech signal may be assumed Toeplitz. Thus, the autocorrelation function of the clean signal in each analysis frame must be estimated. The Fourier transform of a windowed autocorrelation function estimate provides estimates of the variances of the clean signal spectral components.

© 2006 by Taylor & Francis Group, LLC

For a wide-sense stationary noise process, the noise autocorrelation function can be estimated from an initial segment of the noisy signal that contains noise only. If the noise process is not wide-sense stationary, frames of the noisy signal must be classified as in Eq. (20.19), and the autocorrelation function of the noise process must be updated whenever a new noise frame is detected.

In the signal subspace approach [7], the power spectral densities of the noise and noisy processes are first estimated using windowed autocorrelation function estimates. Then, the power spectral density of the clean signal is estimated using Eq. (20.9). That estimate is inverse Fourier transformed to produce an estimate of the desired autocorrelation function of the clean signal. In implementing the MMSE spectral estimator as in Eq. (20.15), a recursive estimator for the variance of each spectral component of the clean signal developed in [4] is often used, see, e.g., [1, 2]. Let $\hat{A}_y(t)$ denote an estimator of the magnitude of a spectral component of the clean signal in a frame at time t. This estimator may be the MMSE estimator as in Eq. (20.15). Let $A_z(t)$ denote the magnitude of the corresponding spectral component of the noisy signal. Let $\widehat{\sigma_w^2}(t)$ denote the estimated variance of the spectral component of the noise process in that frame. Let $\widehat{\sigma_y^2}(t)$ denote the estimated variance of the spectral component of the clean signal in that frame. This estimator is given by

$$\widehat{\sigma_y^2}(t) = \beta \hat{A}_y^2(t-1) + (1-\beta)\max\{A_z^2(t) - \sigma_w^2(t), 0\} \tag{20.24}$$

where $0 \leq \beta \leq 1$ is an experimental constant. We note that while this estimator was found useful in practice, it is heuristically motivated and its analytical performance is not known.

A parametric approach for estimating the power spectral density of the speech signal from a given sample function of the noisy signal was developed by Musicus and Lim [18]. The clean signal in a given frame is assumed a zero mean Gaussian autoregressive process. The noise is assumed a zero mean white Gaussian process that is independent of the signal. The parameter of the noisy signal comprises the autoregressive coefficients, the gain of the autoregressive process, and the variance of the noise process. This parameter is estimated in the ML sense using the EM algorithm. The EM algorithm and conditions for its convergence were originally developed for this problem by Musicus [17]. The approach starts with an initial estimate of the parameter. This estimate is used to calculate the conditional mean estimates of the clean signal and of the sample covariance matrix of the clean signal given the noisy signal. These estimates are then used to produce a new parameter, and the process is repeated until a fixed point is reached or a stopping criterion is met.

This EM procedure is summarized as follows: Consider a scalar autoregressive process $\{Y_t, t = 1, 2, \ldots\}$ of order r and coefficients $a = (a_1, \ldots, a_r)'$. Let $\{V_t\}$ denote an independent identically distributed (iid) sequence of zero mean unit variance Gaussian random variables. Let σ_v denote a gain factor. A sample function of the autoregressive process is given by

$$y_t = -\sum_{i=1}^{r} a_i y_{t-i} + \sigma_v v_t \tag{20.25}$$

Assume that the initial conditions of Eq. (20.25) are zero, i.e., $y_t = 0$ for $t < 1$. Let $\{W_t, t = 1, 2, \ldots\}$ denote the noise process which comprises an iid sequence of zero mean unit variance Gaussian random variables. Consider a k-dimensional vector of the noisy autoregressive process. Let $Y = (Y_1, \ldots, Y_k)'$, and define similarly the vectors V and W corresponding to the processes $\{V_t\}$ and $\{W_t\}$, respectively. Let A denote a lower triangular Toeplitz matrix with the first column given by $(1, a_1, \ldots, a_r, 0, \ldots, 0)'$. Then

$$Z = \sigma_v A^{-1} V + \sigma_w W \tag{20.26}$$

Suppose $\phi_m = (a(m), \sigma_v(m), \sigma_w(m))$ denotes an estimate of the parameter of Z at the end of the mth iteration. Let $A(m)$ denote the matrix A constructed from the vector $a(m)$. Let $R_y(m) = \sigma_v^2(m) A(m)^{-1} (A(m))^{-1})'$ denote the covariance matrix of the autoregressive process of Eq. (20.25) as obtained from ϕ_m. Let $R_z(m) = R_y(m) + \sigma_w^2(m) I$ denote the covariance matrix of the noisy signal as shown in Eq. (20.26) based on ϕ_m. Let $\hat{Y}$ and $\hat{R}$ denote, respectively, the conditional mean estimates of the clean signal Y and of the sample covariance of Y given Z and the current estimate ϕ_m. Under the Gaussian assumptions of

© 2006 by Taylor & Francis Group, LLC

this problem, we have

$$\begin{aligned}\hat{Y} &= E\{Y|Z;\phi_m\} \\ &= R_y(m)R_z^{-1}(m)Z \end{aligned} \tag{20.27}$$

$$\begin{aligned}\hat{R} &= E\{YY'|Z;\phi_m\} \\ &= R_y(m) - R_y(m)R_z^{-1}(m)R_y(m) + \hat{Y}\hat{Y}' \end{aligned} \tag{20.28}$$

The estimation of the statistics of the clean signal in Eq. (20.27) and Eq. (20.28) comprise the E-step of the EM algorithm.

Define the $r+1 \times r+1$ covariance matrix S with entries

$$S(i,j) = \sum_{l=\max(i,j)}^{k-1} \hat{R}(l-i, l-j), \quad i,j = 0,\ldots,r \tag{20.29}$$

Define the $r \times 1$ vector $q = (S(1,0),\ldots,S(r,0))'$ and the $r \times r$ covariance matrix $Q = \{S(i,j), i,j = 1,\ldots,r\}$. The new estimate of parameter ϕ at the end of the $m+1$ iteration is given by

$$a(m+1) = -Q^{-1}q \tag{20.30}$$

$$\sigma_v^2(m+1) = (S(0,0) + 2a'(m)q + a'(m)Qa(m))/k \tag{20.31}$$

$$\begin{aligned}\sigma_w^2(m+1) &= \text{tr}\,(E\{(Z-Y)(Z-Y)'|Z,\phi_m\})/k \\ &= \text{tr}\,(ZZ' - 2\hat{Y}Z' + \hat{R})/k. \end{aligned} \tag{20.32}$$

The calculation of the parameter of the noisy signal in Eq. (20.30) to Eq. (20.32) comprise the M-step of the EM algorithm.

Note that the enhanced signal can be obtained as a by-product of this algorithm using the estimator as in Eq. (20.27) at the last iteration of the algorithm. Formulation of the above parameter estimation problem in a state space assuming colored noise in the form of another autoregressive process, and implementation of the estimators as given in Eq. (20.27) and Eq. (20.28) using Kalman filtering and smoothing was done in [9].

The EM approach presented above differs from an earlier approach pioneered by Lim and Oppenheim [12]. Assume for this comparison that the noise variance is known. In the EM approach, the goal is to derive the ML estimate of the true parameter of the autoregressive process. This is done by local maximization of the likelihood function of the noisy signal over the parameter set of the process. The EM approach results in alternate estimation of the clean signal Y and its sample correlation matrix YY', and of the parameter of the autoregressive process (a, σ_v). The approach taken in [12] aims at estimating the parameter of the autoregressive process that maximizes the joint likelihood function of the clean and noisy signals. Using alternate maximization of the joint likelihood function, the approach resulted in alternate estimation of the clean signal Y and of the parameter of the autoregressive process (a, σ_v). Thus, the main difference between the two approaches is in the estimated statistics from which the autoregressive parameter is estimated in each iteration. This difference impacts the convergence properties of the algorithm [12] which is known to produce an inconsistent parameter estimate. The algorithm [12] is simpler to implement than the EM approach and it is popular among some authors. To overcome the inconsistency, they developed a set of constraints for the parameter estimate in each iteration.

20.6 Concluding Remarks

We have surveyed some aspects of the speech enhancement problem and presented state-of-the-art solutions to this problem. In particular, we have identified the difficulties inherent to speech enhancement, and presented statistical models and distortion measures commonly used in designing estimators for the clean signal. We have mainly focused on speech signals degraded by additive noncorrelated wide-band noise. As we have noted earlier, even for this case, a universally accepted solution is not available and more research

© 2006 by Taylor & Francis Group, LLC

is required to refine current approaches or alternatively develop new ones. Other noise sources, such as room reverberation noise, present a more significant challenge as the noise is a non-stationary process that is correlated with the signal and it cannot be easily modeled. The speech enhancement problem is expected to attract significant research effort in the future due to challenges that this problem poses, the numerous potential applications, and future advances in computing devices.

References

Cappé, O., Elimination of the musical noise phenomenon with the Ephraim and Malah noise suppressor, *IEEE Trans. Speech and Audio Proc.*, vol. 2, pp. 345–349, Apr. 1994.

Cohen, I. and Berdugo, B.H., Speech enhancement for non-stationary noise environments, *Signal Processing*, vol. 81, pp. 2403–2418, 2001.

Drucker, H., Speech processing in a high ambient noise environment, *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 165–168, Jun. 1968.

Ephraim, Y. and Malah, D., Speech enhancement using a minimum mean square error short time spectral amplitude estimator, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1109–1121, Dec. 1984.

Ephraim Y. and Malah, D., Speech enhancement using a minimum mean square error Log-spectral amplitude estimator, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 443–445, Apr. 1985.

Ephraim, Y., Statistical model based speech enhancement systems, *Proc. IEEE*, vol. 80, pp. 1526–1555, Oct. 1992.

Ephraim, Y. and Van Trees, H.L., A signal subspace approach for speech enhancement, *IEEE Trans. Speech and Audio Proc.*, vol. 3, pp. 251–266, July 1995.

Ephraim, Y. and Merhav, N., Hidden Markov Processes, *IEEE Trans. Inform. Theory*, vol. 48, pp. 1518–1569, June 2002.

Gannot, S., Burshtein, D., and Weinstein, E., Iterative and sequential Kalman filter-based speech enhancement algorithms, *IEEE Trans. Speech and Audio Proc.*, vol. 6, pp. 373–385, July 1998.

Gray, R.M., *Toeplitz and Circulant Matrices: II.* Stanford Electron. Lab., Tech. Rep. 6504–1, Apr. 1977.

Lev-Ari, H. and Ephraim, Y., Extension of the signal subspace speech enhancement approach to colored noise, *IEEE Sig. Proc. Let.*, vol. 10, pp. 104–106, Apr. 2003.

Lim, J.S. and Oppenheim, A.V., All-pole modeling of degraded speech, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP–26, pp. 197–210, June 1978.

Lim J.S. and Oppenheim, A.V., Enhancement and bandwidth compression of noisy speech, *Proc. IEEE*, vol. 67, pp. 1586–1604, Dec. 1979.

Lim, J.S., ed., *Speech Enhancement.* Prentice-Hall, New Jersey, 1983.

Makhoul, J., Crystal, T.H., Green, D.M., Hogan, D., McAulay, R.J., Pisoni, D.B., Sorkin, R.D., and Stockham, T.G., *Removal of Noise From Noise-Degraded Speech Signals.* Panel on removal of noise from a speech/noise National Research Council, National Academy Press, Washington, D.C., 1989.

McAulay, R.J. and Malpass, M.L., Speech enhancement using a soft-decision noise suppression filter, *IEEE Trans. Acoust., Speech, Signal Processing,* ASSP-28, pp. 137–145, Apr. 1980.

Musicus, B.R., *An Iterative Technique for Maximum Likelihood Parameter Estimation on Noisy Data.* Thesis, S.M., M.I.T., Cambridge, MA, 1979.

Musicus, B.R. and Lim, J.S., Maximum likelihood parameter estimation of noisy data, *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing*, pp. 224–227, 1979.

Priestley, M.B., *Spectral Analysis and Time Series*, Academic Press, London, 1989.

Quatieri, T.F. and McAulay, R.J., Noise reduction using a soft-decision sin-wave vector quantizer, *IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 821–824, 1990.

Defining Terms

Speech Enhancement: The action of improving perceptual aspects of a given sample of speech signals.
Quality: A subjective measure of the way a speech signal is perceived.

© 2006 by Taylor & Francis Group, LLC

Intelligibility: An objective measure which indicates the percentage of words from a given text that are expected to be correctly understood by listeners.

Signal estimator: A function of the observed noisy signal which approximates the clean signal.

Expectation-maximization: An iterative approach for parameter estimation using alternate estimation and maximization steps.

Autoregressive process: A random process obtained by passing white noise through an all-pole filter.

Wide-sense stationarity: A property of a random process whose second-order statistics do not change with time.

Asymptotically weakly stationarity: A property of a random process indicating eventual wide-sense stationarity.

Hidden Markov Process: A Markov chain observed through a noisy channel.

© 2006 by Taylor & Francis Group, LLC

21

Ad Hoc Networks

Michel D. Yacoub
Paulo Cardieri
Élvio João Leonardo
Álvaro Augusto Machado Medeiros

21.1 Introduction **21**-1
21.2 Routing Algorithms **21**-1
Proactive Algorithms • Reactive Algorithms • Hybrid Algorithms
21.3 Medium Access Protocols **21**-6
Protocols Categories • Industry Standard Protocols • Other Protocols • Comments
21.4 TCP over Ad Hoc Networks **21**-13
Physical Layer Impact • MAC Layer Impact • Mobility Impact • Main TCP Schemes Proposals for Ad Hoc Networks
21.5 Capacity of Ad Hoc Networks **21**-16
Case Studies on Capacity of Ad Hoc Networks • Increasing the Capacity of Ad Hoc Networks

21.1 Introduction

An ad hoc network is a wireless network that is established without the aid of infrastructure or centralized administration. It is formed by a group of wireless terminals (nodes) such that a communication between any two terminals is carried out by means of a store-and-relay mechanism. A terminal wishing to transmit accesses the medium and sends its information to a nearby terminal. Upon receiving such information this terminal determines that this is not addressed to it. It then stores the information in order to relay it to another terminal at an appropriate time, and this process continues until the destination is reached. Note that in ad hoc networks there are no fixed routers. Nodes may be mobile and can be connected dynamically in an arbitrary manner. Nodes function as routers, which discover and maintain routes to other nodes in the network. Ad hoc networks find applications in emergency-and-rescue operations, meeting or conventions, data acquisition operations in inhospitable terrain, sensor networks, and home and office networks. Cheaper hardware, smaller transceivers, and faster processors fuel the increased interest in wireless ad hoc networks. This chapter addresses the ad hoc networks from four main aspects: routing, medium access, TCP/IP issues, and capacity. In routing, the main routing algorithms are illustrated. In medium access, the main medium access protocols are described. In TCP/IP issues, the aspects concerning the performance of TCP/IP in an ad hoc network is discussed. In capacity, some formulation concerning the capacity of the network is tackled.

21.2 Routing Algorithms

The design of routing algorithms in ad hoc networks is a challenging task. Algorithms must provide for a high degree of sophistication and intelligence so that the limited resources inherent to the wireless systems can be dealt with efficiently. They must be robust in order to cope with the unkind wireless environment. At the same time they must be flexible in order to adapt to the changing network conditions such as network size, traffic distribution, and mobility. Routing algorithms have long been used in wired systems

© 2006 by Taylor & Francis Group, LLC

and they are usually classified into two categories: Distant vector (DV) and Link-state (LS). DV algorithms provide each node with a vector containing the hop distance and the next hop to all the destinations. LS algorithms provide each node with an updated view of the network topology by periodical flooding of link information about its neighbors. A direct application of these algorithms in a wireless and mobile environment may be cumbersome. DV protocols suffer from slow route convergence and may create loops. LS protocols require the frequent use of the resources, thence large bandwidth, in order to maintain the nodes updated.

With the increasing interest in wireless networks, a variety of routing algorithms overcoming the limitations of the DV and LS protocols have been proposed. They are usually classified into three categories: proactive or table-driven, reactive or on-demand, and hybrid. The proactive protocols require the nodes to keep tables with routing information. Updates occur on a periodical basis or as soon as changes in the network topology are perceived. The algorithms differ basically in the type of information is kept and in the way updates occur. The reactive protocols create routes on demand. This is accomplished by means of a route discovery process, which is completed once a route has been found or all possible route permutations have been examined. The discovery process occurs by flooding route request packets through the network. After establishing a route, it is maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. The hybrid protocols are both proactive and reactive. Nodes with close proximity form a backbone within which proactive protocols are applied. Routes to faraway nodes are found through reactive protocols.

Proactive Algorithms

CGSR—Clusterhead-gateway switch routing (Chiang, 1997). In CGSR, the whole network is partitioned into clusters of nodes. Within each cluster a *clusterhead* is elected among the nodes. A node may belong to one cluster only, in which case it is an internal node, or to more than one cluster, in which case it becomes a *gateway*. Packets are transmitted from the node to the clusterhead and from the clusterhead to the node. Routing within such a network occurs as follows. Assume source and destination belonging to different clusters. The source sends its packets to the clusterhead, which relays these packets to a gateway, which relays them to another clusterhead, and this process continues until the destination is reached.

DREAM—Distance routing effect algorithm for mobility (Basagni, 1998). In DREAM, GPS is used so that each node can maintain a location table with records of locations of all the nodes. The nodes broadcast control packets for location updating purposes. A source having packets to send calculates the direction toward the destination. It then selects a set of one-hop neighbors in the respective direction. (If the set is empty, the data are flooded to the whole network.) The data header encloses the respective set and is sent. Those nodes specified in the header are entitled to receive and process the data. All the nodes of the paths repeat this process until the destination is reached. Upon receiving the packets, the destination issues an ACK, which is transmitted to the source using the same algorithm.

DSDV—Destination-sequential distance-vector routing (Perkins, 1994). In DSDV, each node keeps a routing table containing all of the possible destinations within the network in conjunction with the number of hops to each destination. The entries are marked with a sequence number assigned by the destination node so that mobile nodes can distinguish stable routes from the new ones in order to avoid routing loops. Table consistency is kept by periodical updates transmitted throughout the network.

FSLS—Fuzzy sighted link state (Santivanez, 2001). In FSLS, an optimal link state update algorithm (hazy sighted link state) is used. Updates occur every $2^k T$, in which k is the hop distance, T is the minimum link state update transmission period, and 2^k is the number of nodes to be updated. FSLS operates in a way very similar to FSR, as described below.

FSR—Fisheye state routing (Iwata, 1999; Pei, 2000). In FSR, each node maintains a topology map, and link state information is exchanged with neighbors periodically. The frequency with which it occurs depends on the hop distance to the current node. Nearby destinations are updated more frequently whereas for the faraway ones the updates are less frequent. Therefore, FSR produces accurate distance and

© 2006 by Taylor & Francis Group, LLC

path information about immediate neighborhood, and imprecise information about the best path to a distant node. On the other hand, such an imprecision is compensated for as the packet approaches its destination.

GSR—Global state routing (Chen, 1998). In GSR, a link state table based on the update messages from neighboring nodes is kept and periodical exchanges of link state information are carried out. The size of update messages increases with the increase of the size of the network, and a considerable amount of bandwidth is required in this case.

HSR—Hierarchical state routing (Pei, 1999). In HSR, the link state algorithm principle is used in conjunction with a hierarchical addressing and topology map. Clustering algorithms may also be used so as to organize the nodes with close proximity into clusters. Each node has a unique identity, which are typically a MAC address and a hierarchical address. A communication between any two nodes occurs by means of physical and logical links. The physical links support the true communication between nodes whereas the logical links are used for the purposes of the hierarchical structure of the communication. This way, several levels in the hierarchy may be built. The lowest level is always the physical level whereas the higher levels constitute the logical levels. Communications then occur starting from the lowest level up to the higher levels and down again to the lowest level.

MMWN–Multimedia support in mobile wireless networks (Kasera, 1997). In MMWN, a clustering hierarchy is used, each cluster having two types of nodes: switch and endpoint. Endpoints do not communicate with each other but with switches only. Within a cluster, one of the switches is chosen as a location manager, which performs location updating and location finding. This means that routing overhead is drastically reduced as compared to the traditional table-driven algorithms. This way, information in MMWN is stored in a dynamically distributed database.

OLSR—Optimized link state routing (Jacquet, 2001). In OLSR, each node keeps topology information on the network by periodically exchanging link state messages. OLSR uses the multipoint replaying strategy (MPR) in order to minimize the size of the control messages and the number of re-broadcasting nodes. To this end, each node selects a set of neighboring nodes (multipoint relays—MPRs) to retransmit its packets. Those not in the selected set may read and process each packet but not retransmit. The selection of the appropriate set is carried out as follows. Each node periodically broadcasts a list of its one-hop neighbors. From the lists the nodes are able to choose a subset of one-hop neighbors that covers all of its two-hop neighbors. An optimal route to every destination is constructed and stored in a routing table.

STAR—Source-tree adaptive routing (Garci-Luna-Aceves, 1999). In STAR, each node keeps a source tree with the preferred paths to destinations. It uses the least overhead routing approach (LORA) to reduce the amount of routing overhead disseminated into the network. The reduction in the amount of messages is achieved by making update dissemination conditional to the occurrence of certain events.

TBRPF—Topology broadcast based on reverse path forwarding (Bellur, 1999; Ogier, 2002). In TBRPF, two separate modules are implemented: neighbor discovery module and the routing module. The first module performs a differential HELLO messages that reports only the changes (up or lost) of neighbors. The second module operates based on partial topology information. The information is obtained through periodic and differential topology updates. If a node n is to send an update message, then every node in the network selects its next hop (parent) node toward that node. Link state updates are propagated in the reverse direction on the spanning tree formed by the minimum-hop paths from all nodes to the sending node. This means that updates originated at n are only accepted if these updates arrive from the respective parent node. They are then propagated toward the children nodes pertaining to n.

WRP—Wireless routing protocol (Murthy, 1996). In WRP, each node maintains a set of tables as follows: distancetable, routingtable, link-costtable, message retransmission list(MRL) table. Each entry of the MRL table contains a sequence number of the update message, a retransmission counter, an acknowledgement required flag vector with one entry per neighbor, and a list of updates sent in the update message. It records which updates need to be retransmitted and which neighbors should acknowledge the retransmission. Update messages are sent only between neighboring nodes and they occur after processing updates or detecting a change in the link to the neighbor. Nodes learn of the existence of their neighbors from the

© 2006 by Taylor & Francis Group, LLC

receipt of ACK and other messages. In case a node is not sending messages of any kind, then a HELLO message is sent within a specified period of time to ensure connectivity.

Reactive Algorithms

ABR—Associativity-based routing (ABR) (Toh, 1996). In ABR, a query-reply technique is used to determine routes to the required destination. Stable routes are chosen based on an associativity tick that each node maintains with its neighbors, with the links with the higher associativity tick being selected. This may not lead to the shortest paths but rather to paths that last longer. In such a case, fewer route reconstructions are needed thence more bandwidth is available. ABR requires periodical beaconing so as to determine the degree of associativity of the links which requires all nodes to remain active at all time, which result in additional power consumption.

AODV—Ad hoc on demand distance vector (Das, 2002). In AODV, periodic beaconing and sequence numbering procedure are used. The packets convey the destination address rather than the full routing information, this also occurring in the route replies. The advantage of AODV is its adaptability to highly dynamic networks. On the other hand, the nodes may experience large delays in route construction.

ARA—Ant-colony-based routing algorithm (Günes, 2002). In ARA, the food searching behavior of ants is used in order to reduce routing over heads. When searching for food, ants leave a trail behind (pheromone) that is followed by the other ants until it vanishes. In the route discovery procedure, ARA propagates a forwarding ANT (FANT) through the network until it reaches the destination. Then a backward ANT (BANT) is returned, a path is created, and data packet dissemination starts. The route is maintained by means of increasing or decreasing the pheromone value at each node. The pheromone at a given node is increased each time a packet travels through it, and it is decreased overtime until it expires. As can be inferred, the size of FANT and BANT is small; therefore the amount of overhead per control packet is minimized.

CBRP—Cluster-based routing protocol (Jiang, 1999). In CBRP, the nodes are grouped into clusters, a cluster presenting a clusterhead. The advantage of using the hierarchical approach is the decrease in the number of control overhead through the network as compared with the traditional flooding methods. Of course, there are overheads associated with the formation and maintenance of the clusters. The long propagation delay due to the hierarchical structure may render the nodes bearing inconsistent topology information, which may lead to temporary routing loops.

DSR—Dynamic source routing (Johnson, 2002). In DSR, there is no periodic beaconing (HELLO message), an important feature that can be used for battery saving purposes, in which the node may enter the sleep mode. Each packet in DSR conveys the full address of the route, and this is a disadvantage for large or highly dynamic networks. On the other hand, the nodes can store multiple routes in their route cache. The advantage of this is that the node can check its route cache for a valid route before initiating route discovery. A valid route found avoids the need for route discovery. And this is an advantageous feature for low mobility networks.

FORP—Flow oriented routing protocol (Su,1999). In FORP, routing fail uredue to mobility is minimized by means of the following algorithm. A Flow-REQ message is disseminated through the network. A node receiving such a message, and based on GPS information, estimates a link expiration time (LET) with the previous hop and append this into its Flow-REQ packet, which is re-transmitted. Upon arrival at the destination, a route expiration time (RET) is estimated using the minimum of all LETs for each node. A Flow-SETUP message is sent back toward the source. Therefore, the destination is able to predict when a link failure may occur. In such a case, a Flow-HANDOFF message is generated and propagated in a similar manner.

LAR—Location aided routing (Ko, 1998). In LAR, using location information routing overhead is minimized, which is commonly present in the traditional flooding algorithms. Assuming each node provided with a GPS, the packets travel in the direction where the relative distance to the destination becomes smaller as they travel from one hop to another.

LMR—Light-weight mobile routing (Corson, 1995). In LMR, the flooding technique is used in order to determine the required routes. Multiple routes are kept at the nodes, the multiplicity being used for

© 2006 by Taylor & Francis Group, LLC

reliability purposes as well as to avoid the re-initiation of a route discovery procedure. In addition, the route information concerns the neighborhood only and not the complete route.

RDMAR—Relative distance micro-discovery ad hoc routing (Aggelou, 1999). In RDMAR, a relative-distance micro-discovery procedure is used in order to minimize routing overheads. This is carried out by means of calculating the distance between source and destination, thence limiting each route request packet to a certain number of hops (i.e., the route discovery procedure becomes confined to localized regions). In fact, this is only feasible if previous communications between source and destination has been established. Otherwise, a flooding procedure is applied.

ROAM—Routing on-demand acyclic multi-path (Raju, 1999). In ROAM, inter-nodal coordination and directed acyclic sub-graphs, derived from the distance of the router to the destination, are used. In case the required destination is no longer reachable multiple flood searches stop. In addition, each time the distance of a router to a destination changes by more than a given threshold, the router broadcasts update messages to its neighboring nodes. This increases the network connectivity at the expense of preventing the nodes entering the sleep mode to save battery.

SSA—Signal stability adaptive (Dube, 1997). In SSA, route selection is carried out based on signal strength and location stability, and not on associativity tick. In addition, route requests sent toward a destination cannot be replied by intermediate nodes, which may cause delays before a route is effectively discovered. This is due to the fact that the destination is responsible for selecting the route for data transfer.

TORA—Temporarily ordered routing algorithm (Park, 1997). In TORA, the key design concept is the localization of control messages to a very small set of nodes near the occurrence of a topological change. The nodes maintain routing information about one-hop nodes. Route creation and route maintenance phases use a height metric to establish a directed acyclic graph rooted at the destination. Then, links are assigned as upstream or downstream based on the relative height metric to neighboring nodes. Route's erasure phase involves flooding a broadcast clear packet throughout the network in order to erase invalid routes.

Hybrid Algorithms

DDR—Distributed dynamic routing (Nikaein, 2001). In DDR, a tree-based routing protocol is used but a root node is not required. The trees are set up by means of periodic beaconing messages, exchanged by neighboring nodes only. Different trees, composing a forest, are connected via gateways nodes. Each tree constitutes a zone, which is assigned a zone ID. The routes are determined by hybrid ad hoc protocols.

DST—Distributed spanning trees based routing protocol (Radhakrishnan, 1999). In DST, all nodes are grouped into trees, within which a node becomes a routing node or an internal node. The root, which is also a node, controls the structure of the tree. This may become a disadvantage of DST for the root node creates a single point of failure.

SLURP—Scalable location update routing protocol (Woo, 2001). In SLURP, the nodes are organized into nonoverlapping zones and a home region for each node in the network is assigned. The home region for each node is determined by means of a static mapping function known to all nodes whose inputs are the node ID and the number of nodes. Thus, all nodes are able to determine the home region for each node. The current location of the node within its home region is maintained by unicasting a location update packet toward its region. Once it reaches its home region, it is broadcast to all nodes within its home.

ZHLS—Zone-based hierarchical link state (Joa-Ng, 1999). In ZHLS, hierarchical topology is used such that two levels are established: node level and zone level, for which the use of GPS is required. Each node then has a node ID and a zone ID. In case a route to a node within another zone is required, the source node broadcasts a zone-level location request to all of the other zones. This generates lower overhead as compared to the flooding approach in reactive protocols. Mobility within its own zone maintains the topology of the network such that no further location search is required. In ZHLS, all nodes are supposed to have a pre-programmed static zone map for initial operation.

ZRP—Zone routing protocol (Haas, 1999). In ZRP, a routing zone is established that defines a range in hops within which network connectivity is proactively maintained. This way, nodes within such a zone

© 2006 by Taylor & Francis Group, LLC

have the routes available immediately. Outside the zone, routes are determined reactively (on demand) and any reactive algorithm may be used.

21.3 Medium Access Protocols

Medium access control (MAC) for wireless ad hoc networks is currently a very active research topic. The characteristics of the network, the diverse physical-layer technologies available, and the range of services envisioned render a difficult task the design of an algorithm to discipline the access to the shared medium that results efficient, fair, power consumption sensitive, and delay bound. A number of issues distinguish wireless MAC protocols from those used in wireline networks (Chandra, 2002), as quoted next.

Half-duplex operation. Due to self-interference (i.e., the energy from the transmitter that leaks into the receiver), it is difficult to construct terminals able to receive while transmitting. Therefore collisiondetection while sending data is not possible and Ethernet-like protocols cannot be used. Since collisions cannot be detected, wireless MAC protocols use collision avoidance mechanisms to minimize the probability of collision.

Time varying channel. In multipath fading channels, the received signal is the sum of time-shifted and attenuated copies of the transmitted signal. With the change of the channel characteristics as well as in the relative position of terminals, the signal envelope varies as a function of time. The signal experiences fading that may be severe. The nodes establishing a wireless link need to sense the channel so as to assess the communication link conditions.

Burst channel errors. Wireless channels experience higher bit error rate than wireline transmissions. Besides, errors occur in bursts as the signal fades, resulting in high probability of packet loss. Therefore, an acknowledgement mechanism must be implemented so that the packet retransmission may be possible in case of packet loss.

Location-dependent carrier sensing. Because the signal strength decays with distance according to a power law, only nodes within a specific range are able to communicate. This gives rise to the hidden and exposed terminals and the capture effect, as described next.

Hidden terminal. Refer to Fig. 21.1 where the relative positions of terminals A, B, and C are shown. B is within range of both A and C but A and C are out of range of each other. If terminal A is transmitting to B and terminal C wishes to transmit to B, it incorrectly senses that the channel is free because it is out of range of A, the current transmitter. If C starts transmitting it interferes with the reception at B. In this case C is termed the hidden terminal to A. The hidden terminal problem can be minimized with the use of the request-to-sent/clear-to-send (RTS/CTS) handshake protocol (to be explained later) before the data transmission starts.

Exposed terminal. An exposed terminal is one that is within range of the transmitter but out of range of the receiver. In Fig. 21.1, if terminal B is transmitting to A and terminal C senses the channel it perceives

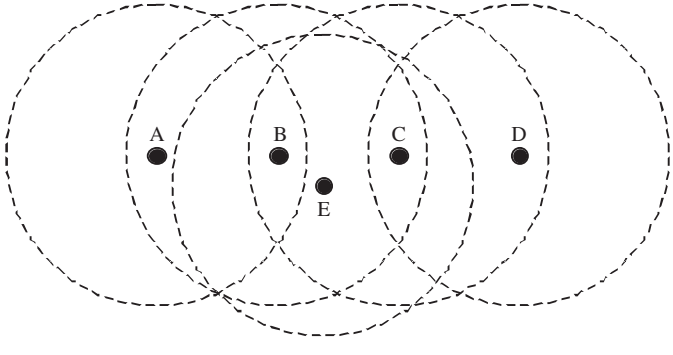


FIGURE 21.1 Hidden-exposed terminal problem. .

© 2006 by Taylor & Francis Group, LLC

it as busy. However, since it is out of range of terminal A it cannot interfere with the current conversation. Therefore it can utilize the channel to establish a parallel link with another terminal that is out of range of B, for instance, terminal D. In this case C is termed the exposed terminal to B. Exposed terminals may result in under-usage of the channel. As in the hidden terminal problem, this also can be minimized with the use of the RTS/CTS handshake protocol.

Capture. Capture at a given terminal occurs in case among several simultaneous signals arriving at it the signal strength of one of them prevails over all of the others combined. In Fig. 21.1, terminals C and E are both within range of terminal B. If C and E are transmitting the interference may result in a collision at B. However, B may be able to receive successfully if one of the signals is much higher than the other, for instance, the signal from E. Capture can improve throughput because it results in less collisions. However, it favors senders that are closer to the intended destination, which may cause unfair allocation of the channel.

From the above considerations, it is promptly inferred that the design of a MAC protocol for ad hoc networks requires a different set of parameters must be considered as compared with those of the wireline systems.

Protocols Categories

Jurdak et al. (Jurdak, 2004), after conducting a survey and analysis of a number of current MAC protocol proposals, offer a set of key features that may be used in order to classify MAC protocols for ad hoc networks.

Channel separation and access. The way the medium is organized is an important issue in the protocol design. For instance, all stations may share a single channel, which they use for control and data transmissions. On the other hand, the medium may be divided into multiple channels, in general one for control and the others for data. The single channel approach was favored in earlier MAC designs because of its simplicity. However, it is intrinsically subject to collisions and it does not perform well in medium to heavy traffic conditions. Particularly at heavy loads, simulations show that single channel protocols are prone to increased number of collisions of control packets, for example, RTS and CTS, which cause increased back off delays while the medium is idle (Tseng, 2002). The choice for multiple channels brings the issue of how to separate these channels. The most common ways of separating channels make use of FDMA, TDMA, and CDMA technologies. *Frequency division multiple access* (FDMA) uses multiple carriers to divide the medium into several frequency slots. It allows multiple transmissions to occur simultaneously although each sender can use only the bandwidth of its assigned frequency slot. *Time division multiple access* (TDMA) divides the medium into fixed length time slots. A group of slots forms a time frame and defines the slot repetition rate. Because of its periodic nature, TDMA protocols are suitable to delay sensitive traffic. In TDMA, a sender uses the whole available bandwidth for the duration of a slot assigned to it. In addition, to access the medium terminals need to keep track of frames and slots and, as a result, TDMA protocols require synchronization among terminals. *Code division multiple access* (CDMA) allows senders to use the whole available bandwidth all the time. Each sender is assigned one of several orthogonal codes and simultaneous transmissions are possible for users are identified by their unique code. A general requirement in CDMA is for power control. The reason behind it is that an unwanted signal that is stronger than the desired signal may overwhelm it at the receiver's antenna. This is known as the near-far effect. *space division multiple access* (SDMA), similarly to CDMA, aims at allowing senders to use the whole available bandwidth all the time. However, the terminals use directional antennas and are allowed to start transmission only if the desired transmission's direction does not interfere with an ongoing conversation.

RTS/CTShandshake. Many MAC protocols for adhoc networks use variants of the RTS/CTS handshake. The original three-way handshake minimizes both the hidden and exposed terminal problems. A terminal wishing to send data first senses the channel. If the channel is idle for the appropriate amount of time, the terminal sends a short request-to-send (RTS) packet. All terminals on hearing the RTS defer their transmissions. The destination responds with a clear-to-send (CTS) packet. All terminals on hearing the

© 2006 by Taylor & Francis Group, LLC

CTS also defer their transmissions. The sender, on receiving the CTS assumes the channel is acquired and initiates the data transmission.

Topology. Ad hoc networks have a large degree of flexibility and uncertainty. Terminals may be mobile and have distinct capabilities and resources. The network must take this into account and adapt dynamically while optimizing performance and minimizing power consumption (Jurdak, 2004). A network topology can be centralized, clustered, or flat. *Centralized topologies* have a single terminal or base station that controls and manages the network. The central terminal may be responsible for broadcasting information relevant to the operation of the network. In addition, terminals may only communicate through the central terminal. *Clustered topologies* create a local version of a centralized network where one terminal assumes some or all of the duties expected from the central terminal. *Flat topologies* implement a fully distributed approach where all terminals are at the same level, and central control is not used. Flat topologies are further divided into single-hop and multiple-hop. *Single-hop* assumes that the destination node is within range of the sender. *Multiple-hop* assumes that the destination node may be beyond the sender's reachable neighbors. In this case, intermediate terminals are responsible for relaying the packets until they reach the intended destination. Single-hop protocols are simpler but pose limitations on the size of the network. Multiple-hop adds scalability to the network at the expense of higher complexity.

Power. Power consumption is a relevant issue for all wireless networks. Power conservation is particularly influential for the mobile terminals because of the limited battery power available. An efficient power conservation strategy involves several aspects. The energy used to transmit the signal represents a large share of the power consumption. Ideally the transmit power used should be just enough to reach the intended destination. Another source of wasted energy is the long periods of time terminals need to spend sensing the channel or overhearing irrelevant conversation. If terminals are able to learn in advance about when the medium will be unavailable they may decide to go into a sleep mode for that period of time in order to save energy. The network behavior may be influenced by the terminals' battery power level, for instance, in the selection of a cluster head or in assigning transmission priorities. Terminals aware of their battery level may adjust their behavior accordingly. The exchange of control messages before the data transmission phase also represents power wastage. Reduced control overhead should therefore be pursued for the sake of power efficiency.

Transmission initiation. Intuitively, it is expected that a terminal wishing to start a conversation must initiate the transmission. And, in fact, most of the protocols are organized this way. However, a receiver-initiated protocol may be more suitable to some specialized networks, for instance, a sensor network. In receiver-initiated protocols the receiver polls its neighbors by sending a ready-to-receive (RTR) packet, which indicates its readiness to receive data. If the receiver is able to know or successfully predict when a neighbor wishes to send its data, this class of protocols actually produces better performance. However, for generalized networks and unpredictable traffic, sender-initiated protocols are still a better choice.

Traffic load and scalability. Protocols are usually optimized for the worst expected scenario. Sparse node distribution and light traffic conditions do not pose a challenge for the implementation of ad hoc networks. The protocols are optimized for high traffic load, high node density, and/or real-time traffic, depending on the intended use. Protocols that offer the possibility of channel reservation are those with best performance on both high load and real-traffic situations. Receiver-initiated approaches also tend to work well in high load conditions because there is a high probability that RTR packets reach terminals wishing to send data. If the network ranks terminals and traffic, then it is able to assign priorities based on the traffic nature. Therefore, it can offer favored handling of real-time traffic. Dense networks tend to suffer from higher interference because of the proximity of transmitting nodes. For this reason the use of power control makes a significant difference in the performance of the network.

Range. Transmission range is the distance from the transmitter's antenna that the radio signal strength still remains above the minimum usable level. Protocols can be classified (Jurdak, 2004) as very short-range (range up to 10 m), short-range (from 10 up to 100 m), medium-range (from 100 up to 1000 m), and long-range (from 1000 m). There is a trade-off between increasing the transmission range and achieving high spatial capacity that needs to be negotiated during the protocol design.

© 2006 by Taylor & Francis Group, LLC

Industry Standard Protocols

IEEE 802.11

The family of IEEE 802.11 standards (IEEE, 1999a; IEEE, 1999b; IEEE, 1999c) can be viewed as a wireless version of the local area network (LAN) protocol Ethernet. The 802.11a standard operates in the unlicensed 5 GHz band and offers data rates up to 54 Mb/s. The commercially popular 802.11b operates in the industrial, scientific, and medical (ISM) band at 2.4 GHz and offers data rates up to 11 Mb/s. The current activity of the 802.11-working group is toward quality of service (QoS) (802.11e, described later) and security (802.11i). The 802.11 standards focus on the specification of the MAC and physical (PHY) layers. While their PHY layers differ, existing 802.11 standards rely on the same medium access mechanisms. The basic (and mandatory) access mechanism is referred to as distributed coordination function (DCF). The optional point coordination function (PCF) is an access mechanism in which a central node (the access point) polls terminals according to a list. DCF is available for both flat ad hoc and centralized topologies whereas PCF is only available in centralized configurations. MAC offers two types of traffic services. The mandatory asynchronous data service is based on the best effort and is suited to delay insensitive data. The optional time-bound service is implemented using PCF.

DCF uses the listen-before-talk scheme based on carrier sense multiple access (CSMA). A terminal wishing to transmit a data packet first monitors the medium activity. If the channel is detected idle the terminal waits for a DCF interframe space (DIFS) time interval (34 us in 802.11a). If the channel remains idle during the DIFS period, the terminal starts transmitting its packet immediately after DIFS has expired. The transmission is successfully completed when the sender receives an acknowledgement (ACK) packet from the destination. However, if the channel is sensed busy a collision avoidance procedure is used. In this procedure, after sensing the channel idle again for a DIFS period, the terminal wishing to transmit waits an additional random backoff time. The terminal then initiates its transmission if the channel remains idle during this additional time. The backoff time is a multiple of the slot time (9 us in 802.11a) and it is determined individually and independently by each station. A random number between zero and contention window (CW) is selected for any new transmission attempt. The back off time is decremented while the medium is in contention phase and frozen otherwise. Thus the backoff time may be carried over for several busy cycles of the medium before it expires. Refer to Fig. 21.2 for an example of the backoff procedure. The initial value for CW is CWmin (15 for 802.11a) and since all terminals operate with the same CWmin value they all have the same initial medium access priority. After any failed transmission, that is, when the transmitted packet is not acknowledged, the sender doubles its CW up to a maximum defined by CWmax (1023 in 802.11a). A now larger CW decreases the probability of collisions if multiple terminals are trying to access the medium. To reduce the hidden terminal problem, 802.11 optionally uses the RTS/CTS handshake. Both RTS and CTS packets include information on how long the data frame transmission is going to last, including the corresponding ACK. Terminals receiving either the RTS or CTS use this information to start a timer, called network allocation vector (NAV), which informs the period of time the medium is unavailable. Between consecutive frames RTS and CTS, and a data frame and its ACK, the short interframe space (SIFS) (16 us in 802.11) is used.

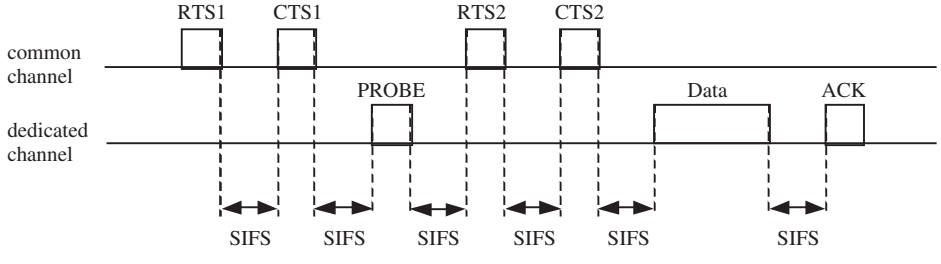


FIGURE 21.2 IEEE 802.11 backoff timing.

© 2006 by Taylor & Francis Group, LLC

SIFS is shorter than DIFS and therefore gives the terminals sending these frames priority to access the medium.

HIPERLAN 1

High Performance LAN type 1 (HIPERLAN 1) is a wireless LAN standard operating in the 5 GHz band, which offers data rate up to 23.5 Mb/s to mobile users in either clustered ad hoc or centralized topology. HIPERLAN 1 offers asynchronous best effort and time-bound services with hierarchical priorities. There are five priority values defined, from zero (highest) to four (lowest). Each individual MAC protocol data unit (PDU) is assigned a priority that is closely related to its normalized residual lifetime (NRL) value. The NRL is an estimation of the time-to-live the PDU has considering the number of hops it still has to travel. A PDU is discarded if its NRL value reaches zero. In addition, some terminals are designed forwarders and are responsible to relay data to distant nodes in a multi-hop fashion. HIPERLAN 1 allows terminals to go into sleep mode in order to save energy. These terminals, called p-savers, inform support terminals, called p-supporters, of their sleep/wake-up patterns. p-supporters then buffer packets directed to p-savers terminals, as required. Although it has some interesting features, HIPERLAN 1 has not been a commercial success. The channel access mechanism used in HIPERLAN 1 is the elimination-yield non-preemptive priority access (EY-NPMA). It comprises three phases: prioritization (determine the highest priority data packets to be sent); contention (eliminate all contenders except one); and transmission. During the prioritization phase, time is divided in five minislots, numbered sequentially from zero to four. A terminal wishing to transmit has to send a burst during the minislot corresponding to its MAC PDU priority. For example, a terminal with a priority two PDU monitors the medium during minislots zero and one before it can assert its intention by transmitting a burst during minislot two. If the medium becomes busy during either minislot zero or one this terminal defers its transmission. Once a burst is transmitted, the prioritization phase ends and only terminals having PDUs at the same priority level remains in the dispute. The contention phase follows. It starts with the contending terminals transmitting an elimination burst. The individual terminals select the burst length, varying from 0 to 12 minislots, at random and independently. After transmitting the burst the terminals sense the medium. If it is busy they defer their transmissions. Otherwise, the remaining terminals enter the yield listening period. They select at random and independently a value between 0 and 9 and start monitoring the medium. If at the end of this period the medium is still idle the terminal assumes it has won the contention and is allowed to transmit its data. Otherwise, it defers its transmission. It is clear that the mechanism does not allow any lower priority packet to be sent if another with higher priority packet is waiting. At the same time the mechanism does not totally eliminate the possibility of collision but reduces it considerably. Similarly to IEEE 802.11, if the medium has been idle for a time longer than the interframe period a terminal wishing to transmit can bypass the EY-NPMA and transmit immediately.

Bluetooth

Bluetooth (Bluetooth, 1999) is a wireless protocol using the license-free ISM band to connect mobile and desktop devices such as computers and computers peripherals, handheld devices, cell phones, etc. The aim is to produce low-cost, low-power, and very-short range devices able to convey voice and data transmissions at a maximum gross rate of 1 Mb/s. Bluetooth uses frequency hopping spread spectrum (FHSS) with 1600 hops/s. For voice, a 64 kb/s full-duplex link called synchronous connection oriented (SCO) is used. SCO assigns a periodic single slot to a point-to-point conversation. Data communication uses the best effort asynchronous connectionless (ACL) link in which up to five slots can be assigned. Terminals in Bluetooth are organized in piconets. A piconet contains one terminal identified as the master and up to seven other active slaves. The master determines the hopping pattern and the other terminals need to synchronize to the piconet master. When it joins a piconet, an active terminal is assigned a unique 3-bit long active member address (AMA). It then stays in either transmit state, when it is engaged in a conversation, or connected state. Bluetooth supports three low-power states: park, hold, and sniff. A parked terminal releases its AMA and is assigned one of the 8-bit long parked member address (PMA). Terminals in the hold and sniff states keep their AMA but have limited participation in the piconet. For instance, a

© 2006 by Taylor & Francis Group, LLC

terminal in the hold state is unable to communicate using ACL. A terminal not participating in any piconet is in stand-by state. Bluetooth piconets can co-exist in space and time and a terminal may belong to several piconets. A piconet is formed when its future master starts an inquiry process, that is, inquiry messages are broadcast in order to find other terminals in the vicinity. After receiving inquiry responses the master may explicitly page terminals to join the piconet. If a master knows already another terminal's identity it may skip the inquiry phase and page the terminal directly. Bluetooth uses time division duplex (TDD) in which master and slave alternate the opportunity to transmit. A slave can only transmit if the master has just transmitted to it, that is, slaves transmit if polled by the master. Transmissions may last one, three, or five slots although only single-slot transmission is a mandatory feature.

IEEE 802.11e

The IEEE 802.11e is an emerging MAC protocol, which defines a set of QoS features to be added to the 802.11 family of wireless LAN standards. Currently there is a draft version of the specifications (IEEE, 2003). The aim is to better serve delay-sensitive applications, such as voice and multi-media. In 802.11e, the contention-based medium access is referred to as enhanced distributed channel access (EDCA). In order to accommodate different traffic priorities, four access categories (AC) have been introduced. To each AC corresponds a backoff entity. The four distinct parallel backoff entities present in each 802.11e terminal are called (from highest to lowest priority): voice, video, best effort, and background. For the sake of comparison, existing 802.11/a/b standards define only one backoff entity per terminal. Each backoff entity has a distinct set of parameters, such as CWmin, Cwmax, and the arbitration interframe space (AIFS). AIFS is at least equal to DIFS and can be enlarged if desired. Another feature added to 802.11e is referred to as transmission opportunity (TxOP). A TxOP defines a time interval, which a back off entity can use to transmit data. It is specified by its starting time and duration, and the maximum length is AC dependent. The protocol also defines the maximum lifetime of each MAC service data unit (MSDU), which is also AC dependent. Once the maximum lifetime has elapsed, the MSDU is discarded. Finally, the protocol allows for the optional block acknowledgement in which a number of consecutive MSDUs are acknowledged with a single ACK frame.

Other Protocols

PRMA—Packet reservation multiple access (Goodman, 1989). In PRMA, the medium is divided into slots and a group of N slots forms a frame. Slots are either reserved or available. The access to the medium is provided by means of the slotted-ALOHA protocol. Data may be either periodic or sporadic, and this is informed in the header of the packet. Terminals are allowed to reserve a slot when they have periodic data to transmit. Once the central node successfully acknowledges the periodic packet, the terminal assumes the slot is reserved and uses it without contention. When the terminal stops sending periodic information then the reserved slot is released. PRMA assumes the existence of a central node but the mechanism can be adapted to other topologies (Jiang, 2002).

MACA-BI—Multiple access with collision avoidance by invitation (Talucci, 1997). In MACA-BI, the receiver polls a prospective sender by transmitting a ready-to-receive (RTR) packet. (This is an example of a receiver-initiated protocol.) In order to perform the polling in a timely fashion the receiver is required to correctly predict the traffic originated by the sender. Periodic traffic makes this task easier. In case either the data buffer or the delay at the terminal increases above a certain threshold this terminal may trigger a conversation by transmitting an RTS packet. Improvements to MACA-BI are proposed in (Garcia, 1999), in which RIMA-SP–receiver initiated multiple access with simple polling –, and RIMA-DP–Receiver initiated multiple access with dual-purpose polling–are introduced. Both protocols render the RTR-data handshake collision free. RIMA-DP gives an additional purpose to the RTR packet: a request for transmission from the polling terminal. After a reservation phase both terminals can exchange data between them.

DBTMA—Dual busy tone multiple access (Haas, 2002). In DBTMA, the RTS/CTS handshake is replaced by two out-of-band busy tones, namely: BTt (transmit busy tone) and BTr (receive busy tone). When a terminal has data to transmit, it first senses the presence of the BTt and BTr tones. If the medium is free

© 2006 by Taylor & Francis Group, LLC

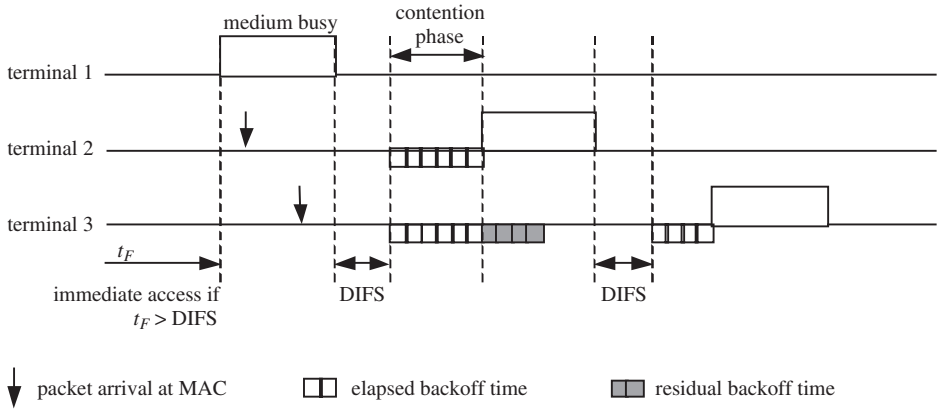


FIGURE 21.3 MAC Handshake Process.

(no busy tone detected), the terminal turns on the BTt, sends an RTS packet, and turns off the BTt. As in other protocols, there is a random backoff time if the medium is busy. The destination terminal, upon receipt of an RTS addressed to it, turns on the BTr and waits for the data. Once the BTr tone is sensed, the sender assumes it has successfully acquired the medium. After waiting a short time (for the BTr to propagate) it transmits the data packet. On successful reception of the data packet the destination terminal turns off the BTr tone, completing the conversation. If no data are received, the BTr tone is turned off after a timer expires at the destination terminal.

Fitzek et al. (Fitzek, 2003) proposes a multi-hop MAC protocol based on the IEEE 802.11. A common channel conveys signaling and dedicated channels carry the data traffic and the ACK packets. Figure 21.3 presents the proposed MAC handshake. The first RTS packet is used to contact the destination and assess its willingness to receive data. The sender includes a list of idle dedicated channels, which is used by the destination terminal to select the dedicated channel. It then transmits this information to the sender in a CTS packet. If no suitable dedicated channel is available the handshake ends. After receiving the CTS packet, the sender transmits a PROBE packet on the dedicated channel. The destination terminal uses this packet to test the channel conditions. It then sends a second CTS packet on the common channel informing about the chosen coding/modulation scheme. The sender to confirm the parameters chosen transmits a second RTS packet. Although at a higher complexity cost, the authors claim that the proposed scheme outperforms the original 802.11.

LA-MAC—Load awareness MAC (Chao, 2003). In LA-MAC, the protocol switches between contention-based and contention-free mechanisms depending on the traffic volume. Contention-based mechanisms are best suited to light traffic conditions in which the probability of collision while attempting to gain the medium is small. For heavy traffic a contention-free mechanism allows higher and more evenly distributed throughput. In (Chao, 2003), the IEEE 802.11 DCF is adopted during contention-based periods while contention-free periods use a token passing protocol. The traffic load is measured by the delay packets are experiencing. Each terminal for the packets it has to transmit computes such delay. During a contention-based period, before a terminal transmits its data packet, it checks the packet's current delay. If the delay is greater than a pre-defined threshold A the terminal creates a token and transmits it attached to the data packet. This indicates to all terminals the start of a contention-free period. Once the delay has fallen below another pre-defined threshold B, the terminal about to transmit removes the token. This indicates the end of the contention-free period and the start of a contention-based period. Threshold A is chosen to be greater than B to give the switching decision some hysteresis.

PCDC—Power controlled dual channel (Muqattash, 2003). In PCDC, the objective is to maintain network connectivity at the lowest possible transmit power. PCDC is a multi-hop protocol that uses the RTS/CTS handshake found in IEEE 802.11 with some modifications. Each terminal is required to keep a

© 2006 by Taylor & Francis Group, LLC

list of neighboring terminals and the transmit power needed to reach them. When a packet is received, the list needs to be visited. If the sender is not known an entry is added. Otherwise, the existing entry is updated. In any case, the receiver needs to re-evaluate its connectivity information and confirm that it knows the cheapest way (in a transmit power sense) to reach all terminals that appears in its neighbor list. For instance, for some terminals it might be cheaper to use an intermediate terminal instead of the direct route. At heavy traffic loads there exist enough packets transiting to keep terminals well informed of their neighborhood. For long idle periods terminals are required to broadcast a "hello" packet periodically for this purpose. PCDC achieves space efficiency and simulations carried by the authors indicate an increase in the network's throughput.

MAC ReSerVation—MAC-RSV (Fang, 2003). In MAC-RSV, a reservation-based multihop MAC scheme is proposed. The TDMA frame consists of data and signaling slots. Data slots are marked as follows: reserved for transmission (RT), reserved for reception (RR), free for transmission (FT), free for reception (FR), or free for transmission and reception (FTR). The signaling slot is divided in minislots with each minislot further divided in three parts: request, reply, and confirm. A terminal wishing to transmit sends an RTS packet. In the RTS, the sender informs its own identity, the intended receiver's identity, and the data slots it wishes to reserve. The intended receiver replies with a CTS if any of the requested slots is among its FR or FTR slots. Otherwise, it remains silent. It is possible that the CTS packet accepts reservation of only a subset of the requested slots. Terminals other than the intended receiver replies with a Not CTS (NCTS) if any of the requested slots is among its RR. Any terminal that detects an RTS collision also replies by sending an NCTS. Otherwise, it remains silent. Finally, if the sender successfully receives a CTS it confirms the reservation by sending a confirm packet (CONF). Otherwise, it remains silent. RTS packets are transmitted in the request part of the minislot; CTS and NCTS use the reply part; and CONF packets use the confirm part. Data slots are divided in three parts: receiver beacon (RB), data, and acknowledgement (ACK). A terminal that has a data slot marked RR transmits an RB with the identity of the active data transmitter. In addition, the receiver acknowledges the correct data reception by transmitting an ACK at the end of the data slot. Simulations carried out by the authors indicate that the proposed protocol outperforms the IEEE 802.11 at moderate to heavy traffic loads.

Comments

In (Jurdak, 2004) a set of guidelines is provided that a suitable general-purpose MAC protocol should follow. In particular, it is mentioned that the use of multiple channels to separate control and data is desirable in order to reduce the probability of collisions. The need of flexible channel bandwidth, multiple channels, and the high bandwidth efficiency suggests that CDMA is the optimal choice for channel partition. Multi-hop support is recommended to ensure scalability with flat or clustered topologies depending on the application. In order to favor power efficient terminals, protocols need to be power aware, must control transmission power, and allow for sleep mode. To complete the set of recommendations, the authors include, for the sake of flexibility, short to medium range networks and a sender-initiated approach.

21.4 TCP over Ad Hoc Networks

TCP is the prevalent transport protocol in the Internet today and its use over ad hoc networks is a certainty. This has motivated a great deal of research efforts aiming not only at evaluating TCP performance over ad hoc networks, but also at proposing appropriate TCP schemes for this kind of networks. TCP was originally designed for wired network, based on the following assumptions, typical of such an environment: packet losses are mainly caused by congestion, links are reliable (very low bit error rate), round-trip times are stable, and bandwidth is constant (Postel, 1981; Huston, 2001). Based on these assumptions, TCP flow control employs a window-based technique, in which the key idea is to probe the network to determine the available resources. The window is adjusted according to an additive-increase/multiplicative-decrease

© 2006 by Taylor & Francis Group, LLC

strategy. When packet loss is detected, the TCP sender retransmits the lost packets and the congestion control mechanisms are invoked, which include exponential backoff of the retransmission timers and reduction of the transmission rate by shrinking the window size. Packet losses are therefore interpreted by TCP as a symptom of congestion (Chandran, 2001). Previous studies on theuse of TCP over cellular wireless networks have shown that this protocol suffers from poor performance mainly because the principal cause of packet loss in wireless networks is no longer congestion, but the error-prone wireless medium (Xylomenos, 2001; Balakrishnan, 1997). In addition, multiple users in a wireless network may share the same medium, rendering the transmission delay time-variant. Therefore, packet loss due to transmission error or a delayed packet can be interpreted by TCP as being caused by congestion. When TCP is used over ad hoc networks, additional problems arise. Unlike cellular networks, where only the last hop is wireless, in ad hoc networks the entire path between the TCP sender and the TCP destination may be made up of wireless hops (multihop). Therefore, as discussed earlier in this chapter, appropriate routing protocols and medium access control mechanisms (at the link control layer) are required to establish a path connecting the sender and the destination. The interaction between TCP and the protocols at the physical, link, and network layers can cause serious performance degradation, as discussed in the following.

Physical Layer Impact

Interference and propagation channel effects are the main causes of high bit error rate in wireless networks. Channel induced errors can corrupt TCP data packets or acknowledgement packets (ACK), resulting in packet losses. If an ACK is not received within the retransmit timeout (RTO) interval, the lost packets may be mistakenly interpreted as a symptom of congestion, causing the invocation of TCP congestion control mechanisms. As a consequence, the TCP transmission rate is drastically reduced, degrading the overall performance. Therefore, the reaction of TCP to packet losses due to errors is clearly inappropriate. One approach to avoid this TCP behavior is to make the wireless channel more reliable by employing appropriate forward error correction coding (FEC), at the expense of a reduction of the effective bandwidth (due to the addition of redundancy) and an increase in the transmission delay (Shakkottai, 2003). In addition to FEC, link layer automatic repeat request (ARQ) schemes can be used to provide faster retransmission than that provided at upper layers. ARQ schemes may increase the transmission delay, leading TCP to assume a large round-trip time or to trigger its own retransmission procedure at the same time (Huston, 2001).

MAC Layer Impact

It is well known that the hidden and exposed terminal problems strongly degrade the overall performance of ad hoc networks. Several techniques for avoiding such problems have been proposed, including the RTS/CTS control packets exchange employed in the IEEE 802.11 MAC protocol. However, despite the use of such techniques, hidden and exposed terminal problems can still occur, causing anomalous TCP behavior. The inappropriate interaction between TCP and link control layer mechanisms in multihop scenarios may cause the so-called TCP instability (Xu, 2001). TCP adaptively controls its transmission rate by adjusting its contention window size. The window size determines the number of packets in flight in the network (i.e., the number of packets that can be transmitted before an ACK is received by the TCP sender). Large window sizes increase the contention level at the link layer, as more packets will be trying to make their way to the destination terminal. This increased contention level leads to packet collisions and causes the exposed terminal problem, preventing intermediate nodes from reaching their adjacent terminals (Xu, 2001). When a terminal cannot deliver its packets to its neighbor, it reports a route failure to the source terminal, which reacts by invoking the route reestablishment mechanisms at the routing protocol. If the route reestablishment takes longer than RTO, the TCP congestion control mechanisms are triggered, shrinking the window size and retransmitting the lost packets. The invocation of congestion control mechanisms results in momentary reduction of TCP throughput, causing the mentioned TCP instability. It has been experimentally verified that reducing the TCP contention window size minimizes

© 2006 by Taylor & Francis Group, LLC

TCP instability (Xu, 2001). However, reduced window size inhibits spatial channel reuse in multihop scenarios. For the case of IEEE 802.11 MAC, which uses a four-way handshake (RTS-CTS-Data-ACK), it can be shown that, in an H-hop chain configuration, a maximum of $^{H}/_{4}$ terminals can simultaneously transmit (Fu, 2003), assuming ideal scheduling and identical packet sizes. Therefore, a window size smaller than this upper limit degrades the channel utilization efficiency. Another important issue related to the interaction between TCP and the link layer protocols regards the unfairness problem when multiple TCP sessions are active. The unfairness problem (Xu, 2001; Tang, 2001) is also rooted in the hidden (collisions) and exposed terminal problems and can completely shut down one of the TCP sessions. When a terminal is not allowed to send its data packet to its neighbor due to collisions or the exposed terminal problem, its backoff scheme is invoked at the link layer level, increasing (though randomly) its backoff time. If the backoff scheme is repeatedly triggered, the terminal will hardly win a contention, and the winner terminal will eventually capture the medium, shutting down the TCP sessions at the loser terminals.

Mobility Impact

Due to terminal mobility, route failures can frequently occur during the lifetime of a TCP session. As discussed above, when a route failure is detected, the routing protocol invokes its route reestablishment mechanisms, and if the discovery of a new route takes longer than RTO, the TCP sender will interpret the route failure as congestion. Consequently, the TCP congestion control is invoked and the lost packets are retransmitted. However, this reaction of TCP in this situation is clearly inappropriate due to several reasons (Chandran, 2001). Firstly, lost packets should not be retransmitted until the route is reestablished. Secondly, when the route is eventually restored, the TCP slow start strategy will force the throughput to be unnecessarily low immediately after the route reestablishment. In addition, if route failures are frequent, TCP throughput will never reach high rates.

Main TCP Schemes Proposals for Ad Hoc Networks

TCP---Feedback

This TCP scheme is based on explicitly informing the TCP sender of a route failure, such that it does not mistakenly invoke the congestion control (Chandran, 2001). When an intermediate terminal detects a route failure, it sends a route failure notification (RFN) to the TCP sender terminal and records this event. Upon receiving an RFN, the TCP sender transitions to a "snooze" state and (i) stops sending packets, (ii) freezes its flow control window size, as well as all its timers, and (iii) starts a route failure timer, among other actions. When an intermediate terminal that forwarded the RFN finds out a new route, it sends a route reestablishment notification (RRN) to the TCP sender, which in turn leaves the snooze state and resumes its normal operation.

TCP with Explicit Link Failure Notification

Explicit link failure notification (ELFN) technique is based on providing TCP sender with information about link or route failures, preventing TCP from reacting to such failures as if congestions had occurred (Holland, 2002). In this approach, the ELFN message is generated by the routing protocol and a notice to TCP sender about link failure is piggybacked on it. When the TCP sender receives this notice, it disables its retransmission timers and periodically probes the network (by sending packets) to check if the route has been reestablished. When an ACK is received, the TCP sender assumes that a new route has been established and resumes its normal operation.

Ad Hoc TCP

A key feature of this approach is that the standard TCP is not modified, but an intermediate layer, called ad hoc TCP (ATCP), is inserted between IP and TCP (transport) layers. Therefore, ATCP is invisible to TCP and terminals with and without ATCP installed can interoperate. ATCP operates based on the network

© 2006 by Taylor & Francis Group, LLC

status information provided by the internet control message protocol (ICMP) and the explicit congestion notification mechanism (ECN) (Floyd, 1994). The ECN mechanism is used to inform the TCP destination of the congestion situation in the network. An ECN bit is included in the TCP header and is set to zero by the TCP sender. Whenever an intermediate router detects congestion, it sets the ECN bit to one. When the TCP destination receives a packet with ECN bit set to one, it informs the TCP sender about the congestion situation, which in turn reduces its transmission rate. ATCP has four possible states: normal, congested, loss, and disconnected. In the normal state ATCP does nothing and is invisible to TCP. In the congested, loss, and disconnected states, ATCP deals with congested network, lossy channel, and partitioned network, respectively. When ATCP sees three duplicate ACKs (likely caused by channel induced errors), ATCP transitions to the loss state and puts TCP into persist mode, ensuring that TCP does not invoke its congestion control mechanisms. In the loss state, ATCP retransmits the unacknowledged segments. When a new ACK arrives, ATCP returns to the normal state and removes TCP from the persist mode, restoring the TCP normal operation. When network congestion occurs, ATCP sees the ECN bit set to one and transitions to congested state. In this state, ATCP does not interfere with TCP congestion control mechanisms. Finally, when a route failure occurs, a destination unreachable message is issued by ICMP. Upon receiving this message, ATCP puts TCP into persist mode and transitions to the disconnected state. While in the persist mode, TCP periodically sends probe packets. When the route is eventually reestablished, TCP is removed from persist mode and ATCP transitions back to the normal state.

21.5 Capacity of Ad Hoc Networks

The classical information theory introduced by Shannon (Shannon, 1948) presents the theoretical results on the channel capacity, that is, how much information can be transmitted over a noisy and limited communication channel. In ad-hoc networks, this problem is led to a higher level of difficulty for the capacity now must be investigated in terms of several transmitters and several receivers. The analysis of the capacity of wireless networks has a similar objective as that of the classical information theory: to estimate the limit of how much information can be transmitted and to determine the optimal operation mode, so that this limit can be achieved. A first attempt to calculate these bounds is made by Gupta and Kumar in (Gupta, 2000). In this work, the authors propose a model for studying the capacity of a static ad hoc network (i.e., nodes do not move), based on the following scenario. Suppose that nodes are located in a region of area 1. Each node can transmit at bits per second over a common wireless channel. Packets are sent from node to node in a multi-hop fashion until their final destination is reached and they can be buffered at intermediate nodes while waiting for transmission. Two types of network configurations are considered: arbitrary networks, where the node locations, traffic destinations, rates, and power level are all arbitrary; and random networks, where the node locations and destinations are random, but they have the same transmit power and data rate. Two models of successful reception over one hop are also proposed:

- Protocol Model—in which a transmission from node i to j, with a distance d_{ij} between them, is successful if $d_{kj} \geq (1 + \Delta)\, d_{ij}$, that is, if the distance between nodes i and j is smaller than that of nodes k and j with both i and k transmitting to j simultaneously over the same channel. The quantity $\Delta > 0$ models the guard zone specified by the protocol to prevent a neighboring node from simultaneous transmission.
- Physical Model—in which, for a subset T of simultaneous transmitting nodes, the transmission from a node $i \in T$ is successfully received by node j if

$$\frac{P_i / d_{ij}^{\alpha}}{N + \sum\limits_{\substack{k \in T \\ k \neq i}} P_k / d_{kj}^{\alpha}} \geq \beta \tag{21.1}$$

  with a minimum signal-to-interference ratio (SIR) β for successful receptions, noise power level N, transmission power level P_i for node i, and signal power decay α.

© 2006 by Taylor & Francis Group, LLC

TABLE 21.1 Capacity Upper Bounds for Arbitrary and Random Networks

	Protocol Model	Physical Model
Arbitrary Networks (transport capacity in bit-meters/s)	$\Theta\left(W\sqrt{n}\right)$	Feasible $cW\sqrt{n}$ Not feasible $c'Wn^{(\frac{\alpha-1}{\alpha})}$ for appropriate values of c and c'
Random Networks (node throughput in bits/s)	$\Theta\left(\frac{W}{\sqrt{n\log n}}\right)$	Feasible $cW/\sqrt{n\log n}$ Not feasible $c'W/\sqrt{n}$ fore appropriate values of c and c'

The transport capacity is so defined as the quantity of bits transported in a certain distance, measured in bit-meter. One bit-meter signifies that one bit has been transported over a distance of one meter toward its destination. With the reception models as described previously, the upper bounds for the transport capacity in arbitrary networks and the throughput per node in random networks were calculated. They are summarized in Table 21.1, where the Knuth's notation has been used, that is, $f(n) = \Theta(g(n))$ denotes that $f(n) = O(g(n))$ as well as $g(n) = O(f(n))$. In Table 21.1, c and c' are constants which are functions of α and β. These results show that, for arbitrary networks, if the transport capacity is divided into equal parts among all nodes, the through put pernode will be $\Theta\ (W/\sqrt{n})$ bitsper second. This shows that, as the number of nodes increases, the throughput capacity for each node diminishes in a square root proportion. The same type of results holds for random networks. These results assume a perfect scheduling algorithm which knows the locations of all nodes and all traffic demands, and which coordinates wireless transmissions temporally and spatially to avoid collisions. Without these assumptions the capacity can be even smaller.

Troumpis and Goldsmith (Toumpis, 2000) extend the analysis of upper limits of (Gupta, 2000) to a three dimensional topology, and incorporated the channel capacity into the link model. In this work, the nodes are assumed as uniformly distributed within a cube of volume $1m^3$. The capacity $C(n)$follows the inequality

$$k_1 \frac{n^{1/3}}{\log(n)} \leq C(n) \leq k_2 \log(n) n^{1/2} \tag{21.2}$$

with probability approaching unity as $n \to \infty$, and k_1, k_2 some positive constants. Equation Eq. (21.2) also suggests that, although the capacity increases with the number of users, the available rate per user decreases.

Case Studies on Capacity of Ad Hoc Networks

IEEE802.11

Li et al (Li, 2001) study the capacity of ad hoc networks through simulations and field tests. Again, the static ad hoc network is the basic scenario, which is justified by the fact that at most mobility scenarios nodes do not move significant distances during packet transmissions. The 802.11 MAC protocol is used to analyze the capacity of different configuration networks.

For the chain of nodes, the ideal capacity is 1/4 of the raw channel bandwidth obtainable from the radio (single-hop throughput). The simulated 802.11-based ad hoc network achieves a capacity of 1/7 of the single-hop throughput, because the 802.11 protocol fails to discover the optimum schedule of transmission and its backoff procedure performs poorly with ad hoc forwarding. The field experiment does not present different results from those obtained in the simulation. The same results are found for the lattice topology. For random networks with random traffic patterns, the 802.11 protocol is less efficient, but the theoretical maximum capacity of $O(1/\sqrt{n})$ per node can be achieved.

It is also shown that the scalability of ad hoc networks is a function of the traffic pattern. In order for the total capacity to scale up with the network size, the average distance between source and destination nodes must remain small as the network grows. Therefore, the key factor deciding whether large networks are feasible is the traffic pattern. For networks with localized traffic the expansion is feasible whereas for networks in which the traffic must traverse it then the expansion is questionable.

© 2006 by Taylor & Francis Group, LLC

Wireless Mesh Networks

A particular case of ad-hoc network, which is drawing significant attention, is the wireless mesh network (WMN). The main characteristic that differentiates a WMN from others ad-hoc networks is the traffic pattern: practically, all traffic is either to or from a node (gateway) that is connected to other networks (e.g. Internet). Consequently, the gateway plays a decisive role in the WMN: the greater the number of gateways the greater the capacity of this network as well as its reliability. Jun and Sichitiu (Jun, (Oct) 2003) analyze the capacity of WMNs with stationary nodes. Their work shows that the capacity of WMNs is extremely dependent on the following aspects:

- Relayed traffic and fairness—Each node in a WMN must transmit relayed traffic as well as its own. Thus, there is an inevitable contention between its own traffic and relayed traffic. In practice, as the offered load at each node increases, the nodes closest to the gateway tends to consume a larger bandwidth, even for a fair MAC layer protocol. The absolute fairness must be forced according to the offered load.
- Nominal capacity of MAC layer (B)—It is defined as the maximum achievable throughput at the MAC layer in one-hop network. It can be calculated as presented in (Jun (April) 2003).
- Link constraints and collision domains—In essence, all MAC protocols are designed to avoid collisions while ensuring that only one node transmits at a time in a given region. The collision domain is the set of links (including the transmitting one) that must be inactive for one link to transmit successfully.

The chain topology is first analyzed. It is observed that the node closer to the gateway has to forward more traffic than nodes farther away. For a n-node network and a per node generated load G, the link between the gateway and the node closer to it has to be able to forward a traffic equal to nG. The link between this node and the next node has to be able to forward traffic equal to $(n-1)G$, and so on. The collision domains are identified and the bottleneck collision domain, which has to transfer the most traffic in the network, is determined. The throughput available for each node is bounded by the nominal capacity B divided by the total traffic of the bottleneck collision domain. The chain topology analysis can be extended to a two-dimensional topology (arbitrary network). The values obtained for the throughput per node are validated with simulation results.

These results lead to an asymptotic throughput per node of $O(1/n)$. This is significantly worse than the results showed in Table 21.1, mainly because of the presence of gateways, which are the network bottlenecks. Clearly, the available throughput improves with the increase of the number of gateways in the network.

Increasing the Capacity of Ad Hoc Networks

The expressions presented in Table 21.1 indicate the best performance achievable considering optimal scheduling, routing, and relaying of packets in the static networks. This is a bad result as far as scalability is concerned and encourages researches to pursue techniques that increase the average throughput. One approach to increase capacity is to add relay-only nodes in the network. The major disadvantage of this scheme is that it requires alarge number of pure relay nodes. For random networks under the protocol model with m additional relay nodes, the throughput available per node becomes $\Theta(W(n+m)/n\sqrt{(n+m)\log(n+m)})$ (Gupta, 2000). For example, in a network with 100 senders, at least 4476 relay nodes are needed to quintuplicate the capacity (Gupta, 2000). Another strategy is to introduce mobility into the model. Grossglauser and Tse (Grossglauser, 2001) show that it is possible for each sender-receiver pair to obtain a constant fraction of the total available bandwidth, which is independent of the number of pairs, at the expense of an increasing delay in the transmission of packets and the size of the buffers needed at the intermediate relay nodes. The same results are presented by Bansal and Liu (Bansal, 2003), but with low delay constraints and a particular mobility model similar to the random waypoint model (Bettstetter, 2002). However, mobility introduces new problems such as maintaining connectivity within the ad hoc network, distributing routing information, and establishing access control. (An analysis on the connectivity of ad hoc networks can be found at (Bettstetter, 2002).) The nodes can also be grouped into small clusters, where in each cluster a specific node (clusterhead) is designated to carry all the relaying packets (Lin, 1997). This can increase the

© 2006 by Taylor & Francis Group, LLC

capacity and reduce the impact of the transmission overhead due to routing and MAC protocols. On the other hand, the mechanisms to update the information in clusterheads generate additional transmissions, which reduces the effective node throughput.

References

Aggelou, G., Tafazolli, R., RDMAR: A Bandwidth-efficient Routing Protocol for Mobile Ad Hoc Networks, *ACM International Workshop on Wireless Mobile Multimedia (WoWMoM)*, 1999, pp. 26–33.

Balakrishnan, H., Padmanabhan, V.N., Seshan, S., Katz, R.H.A, A Comparison of Mechanisms for Improving TCP Performance over Wireless Links, IEEE/ACM Trans. on Networking, vol. 5, No. 6, pp. 756–769, December 1997.

Bansal, N. and Liu, Z. Capacity, Delay and Mobility in Wireless Ad-Hoc Networks, *Proc. IEE Infocom* '03, April 2003.

Basagni, S. et al., A Distance Routing Effect Algorithm for Mobility (DREAM), *ACM/IEEE Int'l. Conf. Mobile Comp. Net.*, pp. 76–84, 1998.

Bellur, B. and Ogier, R.G., A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks, *Proc. IEEE INFOCOM '99*, New York, March 1999.

Bettstetter, C., On the Minimum Node Degree and Connectivity of a Wireless Multihop Network, *Proc. ACM Intern. Symp. On Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2002.

Bluetooth SIG, Specification of the Bluetooth System, vol. 1.0, 1999, available at: http://www.bluetooth.org.

Chandra, A., Gummalla, V., Limb, J.O., Wireless Medium Access Control Protocols, *IEEE Communications Surveys and Tutorials [online]*, vol. 3, no. 2, 2000, available at: http://www.comsoc.org/pubs/surveys/.

Chandran, K., Raghunathan, S., Venkatesan, S., Prakash, R., A Feedback-Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks, *IEEE Personal Communications*, pp. 34–39, February 2001.

Chao, C.M., Sheu, J.P., Chou, I-C., A load awareness medium access control protocol for wireless ad hoc network, *IEEE International Conference on Communications, ICC'03*, vol. 1, 11–15, pp. 438–442, May 2003.

Chatschik, B., An overview of the Bluetooth wireless technology, *IEEE Communications Magazine*, vol. 39, no. 12, pp. 86–94, Dec. 2001.

Chen, T.-W., Gerla, M., Global State Routing: a New Routing Scheme for Ad-hoc Wireless Networks, *Proceedings of the IEEE ICC*, 1998.

Chiang, C.-C. and Gerla, M., Routing and Multicast in Multihop, Mobile Wireless Networks, *Proc. IEEE ICUPC '97*, San Diego, CA, Oct. 1997.

Corson, M.S. and Ephremides, A., A Distributed Routing Algorithm for Mobile Wireless Networks, *ACM/Baltzer Wireless Networks 1*, vol 1, pp. 61-81, 1995.

Das, S., Perkins, C., and Royer, E., Ad Hoc on Demand Distance Vector (AODV) Routing, Internet Draft, draft-ietf-manet-aodv-11.txt, 2002.

Dube, R., Rais, C., Wang, K., and Tripathi, S., Signal Stability Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks, *IEEE Personal Communication 4*, vol 1, pp. 36–45, 1997.

Fang, J.C. and Kondylis, G.D., A synchronous, reservation based medium access control protocol for multihop wireless networks, *2003 IEEE Wireless Communications and Networking, WCNC 2003*, vol. 2, 16–20. March 2003, pp. 994–998.

Fitzek, F.H.P., Angelini, D., Mazzini, G., and Zorzi, M., Design and performance of an enhanced IEEE 802.11 MAC protocol for multihop coverage extension, *IEEE Wireless Communications*, vol. 10, no. 6, pp. 30–39, Dec. 2003.

Floyd, S., TCP and Explicit Congestion Notification, *ACM Computer Communication Review*, vol. 24, pp. 10–23, October 1994.

Fu, Z., Zerfos, P., Luo, H., Lu, S., Zhang, L., and Gerla, M., The Impact of Multihop Wireless Channel on TCP Throughput and Loss, *IEEE INFOCOM*, pp. 1744–1753, 2003.

© 2006 by Taylor & Francis Group, LLC

Garcia-Luna-Aceves, J.J. and Marcelo Spohn, C., Source-tree Routing in Wireless Networks, *Proceedings of the Seventh Annual International Conference on Network Protocols*, Toronto, Canada, p. 273, October 1999.

Garcia-Luna-Aceves, J.J. and Tzamaloukas, A., Reversing the Collision-Avoidance Handhske in Wireless Networks, *ACM/IEEE MobiCom'99*, pp. 15–20, August 1999.

Goodman, D.J., Valenzuela, R.A., Gayliard, K.T., Ramamurthi, B., Packet reservation multiple access for local wireless communications, *IEEE Transactions on Communications*, vol. 37, no. 8, pp. 885–890, Aug. 1989.

Grossglauser, M. and Tse, D., Mobility Increases the Capacity of Ad Hoc Wireless Networks, *Proc. IEE Infocom '01*, April 2001.

Günes, M., Sorges, U., and Bouazizi, I., ARA–The Ant-colony Based Routing Algorithm for Manets, *ICPP Workshop on Ad Hoc Networks (IWAHN 2002)*, pp. 79–85, August 2002.

Gupta, P. and Kumar, P.R., The Capacity of Wireless Networks, *IEEE Trans. Info. Theory*, vol. 46, March 2000.

Haas, Z.J. and Deng, J., Dual busy tone multiple access (DBTMA)—a multiple access control scheme for ad hoc networks, *IEEE Transactions on Communications*, vol. 50, no. 6, pp. 975–985, June 2002.

Haas, Z.J. and Pearlman, R., Zone Routing Protocol for Ad-hoc Networks, *Internet Draft*, draft-ietf-manet-zrp-02.txt,1999.

Holland, G. and Vaidya, N., Analysis of TCP Performance over Mobile Ad Hoc Networks, *Wireless Networks*, Kluwer Academic Publishers, vol. 8, pp. 275–288, 2002.

Huston, G., TCP in a Wireless World, *IEEE Internet Computing*, pp. 82–84, March–April, 2001.

IEEE 802.11 WG, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, *IEEE/ANSI Std.* 802-11, 1999 edn.

IEEE 802.11 WG, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: high-speed physical layer in the 5 GHz band, *IEEE Std*. 802-11a.

IEEE 802.11 WG, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: higher-speed physical layer extension in the 2.4 GHz band, *IEEE Std*. 802-11b.

IEEE 802.11 WG, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), *IEEE Draft Std*. 802.11e/D5.0, August 2003.

Iwata, A. et al., Scalable Routing Strategies for Ad-hoc Wireless Net-works, *IEEE JSAC*, pp. 1369–179, August. 1999.

Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., and Viennot, L., Optimized Link State Routing Protocol for Ad Hoc Networks, *IEEE INMIC*, Pakistan, 2001.

Jiang, M., Ji, J., and Tay, Y.C., Cluster based routing protocol, *Internet Draft*, draft-ietf-manet-cbrp-spec-01.txt, 1999.

Jiang, S., Rao, J., He, D., Ling, X., and Ko, C.C., A simple distributed PRMA for MANETs, *IEEE Transactions on Vehicular Technology*, vol. 51, no. 2, pp. 293–305, March 2002.

Joa-Ng, M. and Lu, I.-T., A Peer-to-peer Zone-based Two-level Link State Routing for Mobile Ad Hoc Networks, *IEEE Journal on Selected Areas in Communications 17*, vol. 8, pp. 1415–1425, 1999.

Johnson, D., Maltz, D., and Jetcheva, J., The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, *Internet Draft*, draft-ietf-manet-dsr-07.txt, 2002.

Jun, J. and Sichitiu, M.L., The Nominal Capacity of Wireless Mesh Networks, *IEEE Wireless Communications*, October 2003.

Jun, J., Peddabachagari, P., and Sichitiu, M.L., Theoretical Maximum Throughput of IEEE 802.11 and its Applications, *Proc. 2nd IEEE Int'l Symp. Net. Comp. and Applications*, pp. 212–25, April 2003.

Jurdak, R., Lopes, C.V., and Baldi, P., A Survey, Classification and Comparative Analysis of Medium Access Control Protocols for Ad Hoc Networks, *IEEE Communications Surveys and Tutorials* [online], vol. 6, no. 1, 2004. available at: http://www.comsoc.org/pubs/surveys/.

© 2006 by Taylor & Francis Group, LLC

Kasera, K.K. and Ramanathan, R., A Location Management Protocol for Hierarchically Organized Multihop Mobile Wireless Networks, *Proceedings of the IEEE ICUPC'97*, San Diego, CA, pp. 158–162, October 1997.

Ko, Y.-B., Vaidya, N.H., Location-aided Routing (LAR) in Mobile Ad Hoc Networks, *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, Dallas, TX, 1998.

Li, J., Blake, C., De Couto, D.S.J., Lee, H.I., and Morris, R., Capacity of Ad Hoc Wireless Networks, *Proc. 7th ACM Int'l Conf. Mobile Comp. and Net.*, pp. 61–69, July 2001.

Lin, C.R. and Gerla, M., Adaptative Clustering for Mobile Wireless Networks, *IEEE Journal on Selected Ares in Communications*, vol. 15, pp. 1265–1275, September 1997.

Mangold, S., Sunghyun Choi, Hiertz, G.R., Klein, O., and Walke, B., Analysis of IEEE 802.11e for QoS support in wireless LANs, *IEEE Wireless Communications*, vol. 10, no. 6, pp. 40–50, December 2003.

Muqattash, A. and Krunz, M., Power controlled dual channel (PCDC) medium access protocol for wireless ad hoc networks, *22nd. Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2003*, vol. 1, pp. 470–480, 30 March–3 April 2003.

Murthy, S. and Garcia-Lunas-Aceves, J.J., An Efficient Routing Protocol for Wireless Networks, *ACM Mobile Networks and App. J.*, Special Issue on Routing in Mobile Communication Networks, pp. 183–197, Oct. 1996.

Nikaein, N., Laboid, H., and Bonnet, C., Distributed Dynamic Routing Algorithm (DDR) for Mobile Ad Hoc Networks, *Proceedings of the MobiHOC 2000:First Annual Workshop on Mobile Ad Hoc Networking and Computing*, 2000.

Ogier, R.G. et al., Topology Broadcast based on Reverse-Path Forwarding (TBRPF), draft-ietf-manet-tbrpf-05.txt, INTERNET-DRAFT, MANET Working Group, March 2002.

Park, V.D. and Corson, M.S., A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, *Proceedings of INFOCOM*, April 1997.

Pei, G., Gerla, M., and Chen, T.-W., Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks, *Proc. ICC 2000*, New Orleans, LA, June 2000.

Pei, G., Gerla, M., Hong, X., and Chiang, C., A Wireless Hierarchical Routing Protocol with Group Mobility, *Proceedings of Wireless Communications and Networking*, New Orleans, 1999.

Perkins, C.E. and Bhagwat, P., Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, *Comp. Commun. Rev.*, pp. 234–244, Oct. 1994.

Postel, J., Transmission Control Protocol, *IETF RFC 793*, September 1981.

Radhakrishnan, S., Rao, N.S.V., Racherla, G., Sekharan, C.N., and Batsell, S.G., DST–A Routing Protocol for Ad Hoc Networks Using Distributed Spanning Trees, *IEEE Wireless Communications and Networking Conference*, New Orleans,1999.

Raju, J. and Garcia-Luna-Aceves, J., A New Approach to On-demand Loop-free Multipath Routing, *Proceedings of the 8th Annual IEEE International Conference on Computer Communications and Networks (ICCCN)*, pp. 522–527, Boston, MA, October 1999.

Santivanez, C., Ramanathan, R., and Stavrakakis, I., Making Link-State Routing Scale for Ad Hoc Networks, *Proc. 2001 ACM Int'l. Symp. Mobile Ad Hoc Net. Comp.*, Long Beach, CA, October 2001.

Schiller, J., Mobile Communications, Addison-Wesley, Reading, MA, 2000.

Shakkottai, S., Rappaport, T.S., and Karlsson, P.C, Cross-Layer Design for Wireless Networks, *IEEE Communications Magazine*, pp. 74–80, October 2003.

Shannon, C.E., A mathematical theory of communication, *Bell System Technical Journal*, vol. 79, pp. 379–423, July 1948.

Su, W. and Gerla, M., IPv6 Flow Handoff in Ad-hoc Wireless Networks Using Mobility Prediction, *IEEE Global Communications Conference*, Rio de Janeiro, Brazil, pp. 271–275, December 1999.

Talucci, F., Gerla, M., and Fratta, L., MACA-BI (MACA By Invitation)-a receiver oriented access protocol for wireless multihop networks, *8th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, PIMRC'97, Vol. 2, pp. 435–439, 1-4 Sept. 1997.

© 2006 by Taylor & Francis Group, LLC

Tang, K., Gerla, M., and Correa, M., Effects of Ad Hoc MAC Layer Medium Access Mechanisms under TCP, *Mobile Networks and Applications*, Kluwer Academic Publishers, Vol. 6, pp. 317–329, 2001.

Toh, C., A Novel Distributed Routing Protocol to Support Ad-hoc Mobile Computing, *IEEE 15th Annual International Phoenix Conf.*, pp. 480–486, 1996.

Toumpis, S. and Goldsmith, A., Ad Hoc Network Capacity, Conference Record of Thirty-fourth Asilomar Conference on Signals Systems and Computers, Vol 2, pp. 1265–1269, 2000.

Tseng, Y.C. and Hsieh, T.Y., Fully power-aware and location-aware protocols for wireless multi-hop ad hoc networks, *11th. International Conference on Computer Communications and Networks*, pp. 608–613, 14–16 Oct. 2002.

Woo, S.-C. and Singh, S., Scalable Routing Protocol for Ad Hoc Networks, *Wireless Networks 7*, Vol. 5, 513–529, 2001.

Xu, S. and Saadawi, T., Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks? *IEEE Communications Magazine*, pp. 130–137, June 2001.

Xylomenos, G., Polyzos, G. C., Mahonen, P., and Saaranen, M., TCP Performance Issues over Wireless Links, *IEEE Communications Magazine*, Vol. 39, No. 4, pp. 53–58, April 2001.

© 2006 by Taylor & Francis Group, LLC

22

Network Communication

22.1 General Principles of Network Analysis and Design . . 22-1
Use of Top-Down Business Oriented Approach • Use of Open Systems Interconnection (OSI) Model • Differentiation Among Major Categories of Networking
22.2 Personal Remote Connectivity 22-5
Applications
22.3 Local Area Networks 22-6
LAN Applications • Local Communication • Local Area Network Hardware • Wiring Centers Technology Analysis
22.4 Internetworking 22-16
Applications: What Can Internetworking Do for You? • Internetworking Technology
22.5 Wide Area Networks 22-23
Applications

James E. Goldman

22.1 General Principles of Network Analysis and Design

Use of Top-Down Business Oriented Approach

Network communication is the transport of data, voice, video, image, or facsimile (fax) from one location to another achieved by compatibly combining elements of hardware, software, and media. From a business perspective, network communications is delivering the right information to the right decision maker at the right place and time for the right cost. Because there are so many variables involved in the analysis, design, and implementation of such networks, a structured methodology must be followed in order to assure that the implemented network meets the communications needs of the intended business, organization, or individual.

One such structured methodology is known as the top-down approach. Such an approach can be graphically illustrated in a top-down model as shown in Fig. 22.1. Using a top-down approach as illustrated in the top-down model is relatively straightforward.

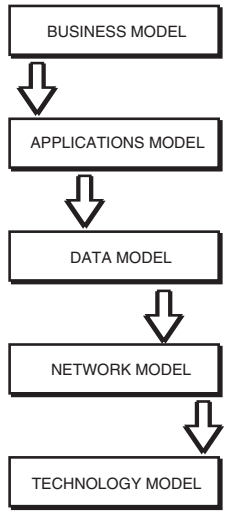


FIGURE 22.1 Top-down design approach: business data comunications analysis.

© 2006 by Taylor & Francis Group, LLC

Data Traffic Analysis Type	Data Traffic Analysis Description
Payload type analysis	Most locations will require at least voice and data service. Videoconferencing and multimedia may need to be supported as well. All payload types should be considered and documented prior to circuit and networking hardware selection.
Transaction analysis	Use process flow analysis or document flow analysis to identify each type of transaction. Analyze detailed data requirements for each transaction type.
Time studies	Once all transaction types have been identified, analyze when and how often each transaction type occurs.
Traffic volume analysis	By combining all types of transactions along with the results of the time study, a time-sensitive traffic volume requirements profile can be produced. This will be a starting point for mapping bandwidth requirements to circuit capacity.
Mission-critical analysis	Results of this analysis may dictate the need for special data security procedures such as encryption or special network design considerations such as redundant links.
Protocol stack analysis	Each corporate location's data traffic is analyzed as to protocols which must be transported across the corporate wide area network. Many alternatives for the transport of numerous protocols exist, but first these protocols must be identified.

FIGURE 22.2 Data traffic analysis.

One must start with the business level objectives. What is the company (organization, individual) trying to accomplish by installing this network? Without a clear understanding of business level objectives, it is nearly impossible to configure and implement a successful network.

Once business level objectives are understood, one must understand the *applications* which will be running on the computer systems attached to these networks. After all, it is the applications that will be generating the traffic that will travel over the implemented network.

Once applications are understood and have been documented, the *data* which those applications generate must be examined. In this case, the term data is used in a general sense as today's networks are likely to transport a variety of payloads including voice, video, image, and fax in addition to true data. Data traffic analysis must determine not only the amount of data to be transported, but also must determine important characteristics about the nature of that data. A summarization of data traffic analysis is outlined in Fig. 22.2. It is also during this stage of the top-down analysis in which the geographic proximity of the nodes of the network are examined.

Geographic proximity is one differentiating factor among different categories of networking, which will be examined further subsequently.

Once data traffic analysis has been completed, the following should be known:

1. Physical locations of data (Where?)
2. Data characteristics and compatibility issues (What?)
3. Amount of data generated and transported (How much?)

Given these requirements as determined by the upper layers of the top-down model, the next job is to determine the requirements of the *network* that will possess the capability to deliver this data in a timely, cost-effective manner. Details on the determination of these requirements comprise the remainder of this section on network communications. These network performance criteria could be referred to as *what* the implemented network must do in order to meet the business objectives outlined at the outset of this top-down analysis. These requirements are also sometimes referred to as the *logical network design*.

The *technology* layer analysis, in contrast, will determine *how* various hardware and software components will be combined to build a functional network which will meet predetermined business objectives. The delineation of required technology is often referred to as the *physical network design.*

© 2006 by Taylor & Francis Group, LLC

LAYER	MAJOR FUNCTIONALITY	BLUEPRINT
7 APPLICATION	Layer where Network Operating System and application programs reside. Layer that the user interfaces with.	FURNISHINGS Chairs, couches, tables, paintings
6 PRESENTATION	Assures reliable session transmission between applications. Takes care of differences; data representation.	INTERIOR CARPENTRY Cabinets, shelves, mouldings
5 SESSION	Enables two applications to communicate across the network.	ELECTRICAL Connection between light switch and light wiring.
4 TRANSPORT	Assures reliable transmission from end-to-end, usually across multiple nodes.	HEATING/COOLING/ PLUMBING Furnace, A/C, ductwork
3 NETWORK	This layer sets up the pathways, or end-to-end connections usually across a long distance, or multiple modes.	STRUCTURAL Studs, dry wall
2 DATA LINK	Puts messages together, attaches proper headers to be sent out or received, assures messages are delivered between two points.	FOUNDATION Concrete support upon which entire structure is built.
1 PHYSICAL	Layer that is concerned with transmitting bits of data over a physical medium.	EXCAVATION Site preparation to allow other phases (layers) to proceed according to plans.

FIGURE 22.3 The OSI reference model.

Overall, the relationship between the layers of the top-down model could be described as follows: analysis at upper layers produces requirements that are passed down to lower layers while solutions meeting these requirements are passed back to upper layers. If this relationship among layers holds true throughout the business oriented network analysis, then the implemented technology (bottom layer) should meet the initially outlined business objectives (top layer).

Use of Open Systems Interconnection (OSI) Model

Determining which technology to employ to meet the requirements determined in the logical network design (network layer) requires a structured methodology of its own. Fortunately, a framework for organizing networking technology solutions has been developed by the International Standards Organization (ISO) and is known as the open systems interconnection (OSI) model. The OSI model is illustrated in Fig. 22.3.

The OSI model divides the communication between any two networked computing devices into seven layers or categories. The OSI model allows data communications technology developers as well as standards developers to talk about the interconnection of two networks or computers in common terms without dealing in proprietary vendor jargon.

These common terms are the result of the layered architecture of the seven-layer OSI model. The architecture breaks the task of two computers communicating to each other into separate but interrelated tasks, each represented by its own layer. As can be seen in Fig. 22.3, the top layer (layer 7) represents the application program running on each computer and is therefore aptly named the *application layer*. The bottom layer (layer 1) is concerned with the actual physical connection of the two computers or networks and is therefore named the *physical layer*. The remaining layers (2–6) may not be as obvious but, nonetheless, represent a sufficiently distinct logical group of functions required to connect two computers, as to justify a separate layer.

To use the OSI model, a network analyst lists the known protocols for each computing device or network node in the proper layer of its own seven-layer OSI model. The collection of these known protocols in

© 2006 by Taylor & Francis Group, LLC

their proper layers in known as the *protocol stack* of the network node. For example, the physical media employed, such as unshielded twisted pair, coaxial cable, or fiber optic cable, would be entered as a layer 1 protocol, whereas **ethernet** or **token ring** network architectures might be entered as a layer 2 protocol. Other examples of possible protocols in respective layers will be explored in the remainder of this section.

The OSI model allows network analysts to produce an accurate inventory of the protocols present on any given network node. This protocol profile represents a unique personality of each network node and gives the network analyst some insight into what *protocol conversion,* if any, may be necessary in order to get any two network nodes to communicate successfully. Ultimately, the OSI model provides a structured methodology for determining what hardware and software technology will be required in the physical network design in order to meet the requirements of the logical network design.

Perhaps the best analogy for the OSI reference model, which illustrates its architectural or framework purpose, is that of a blueprint for a large office building or skyscraper. The various subcontractors on the job may only be concerned with the layer of the plans that outlines their specific job specifications. However, each specific subcontractor needs to be able to depend on the work of the lower layers subcontractors just as the subcontractors of the upper layers depend on these subcontractors performing their function to specification. Similarly, each layer of the OSI model operates independently of all other layers, while depending on neighboring layers to perform according to specification while cooperating in the attainment of the overall task of communication between two computers or networks.

Differentiation Among Major Categories of Networking

As part of the top-down analysis, geographic proximity of computers or network nodes was mentioned as a key piece of analysis information. Although there are no hard and fast rules for network categorization, following are a few of the more common categories of networking:

- Remote connectivity: A single remote user wishes to access local network resources. This type of networking is particularly important to mobile professionals such as sales representatives, service technicians, field auditors, etc.
- Local area networking: Multiple users' computers are interconnecting for the purpose of sharing applications, data, or networked technology such as printers or CD-ROMs. Local area networks (LANs) may have anywhere from two or three users to several hundred. LANs are often limited to a single department or floor in a building, although technically any single location corporation could be networked via a LAN.
- Internetworking: Also known as LAN-to-LAN networking or connectivity, internetworking involves the connection of multiple LANs and is very common in corporations in which users on departmental LANs now need to share data or otherwise communicate. The challenge of internetworking is in getting departmental LANs of different protocol stacks (as determined by use of the OSI model) to talk to each other, while only allowing authorized users access to the internetwork and other LANs. Variations of internetworking also deal with connecting LANs to mainframes or minicomputers rather than to other LANs.
- Wide area networking: Also known as *enterprise networking*, involves the connection of computers, network nodes, or LANs over a sufficient distance as to require the purchase of wide area network (WAN) service from the phone company or alternative carrier. In some cases, the wide area portion of the network may be owned and operated by the corporation itself. Nonetheless, the geographic distance between nodes is the determining factor in categorizing a wide area network. A subset of WANs known as metropolitan area networks (MANs) are confined to a campus or metropolitan area of usually not more than a few miles in diameter.

The important thing to remember is that categorization of networking is somewhat arbitrary and that what really matters is that the proper networking technology (hardware and software) is specified in any given networking opportunity in order to meet stated business objectives.

© 2006 by Taylor & Francis Group, LLC

22.2 Personal Remote Connectivity

Applications

The overall methodology for analysis and design of remote connectivity networking can be summarized as follows:

1. Needs analysis
2. Logical topology choice
3. Physical topology or architecture choice
4. Technology review and specification

Remote Connectivity Needs Analysis

Remote connectivity needs analysis involves documenting the nature and extent of the use of local LAN resources by the remotely connected user. Choices of logical or physical topology for this remote LAN connectivity may be limited depending on the outcome of the remote connectivity needs analysis. Among the possible information sharing needs of remote users are the following: (1) exchange e-mail, (2) upload and download files, (3) run interactive application programs remotely, and (4) utilize LAN attached resources such as printers. One additional question will have a direct impact on topology choice: (5) How many remote users will require simultaneous access to local LAN attached resources?

Remote connectivity architectures comprise the combination of a chosen remote connectivity logical topology and a chosen remote connectivity physical topology.

Logical Topologies

Remote connectivity logical topologies are differentiated by: (1) location of application program execution (local or remote PC) and (2) nature of the data traffic between the local and remote PC. The two most common remote connectivity logical topologies or operations modes are: (1) remote client mode and (2) remote control mode.

Remote Client Mode

Remote client mode, alternatively known as remote access node, executes and often stores applications on the remote PC, using only shared data and other locally attached LAN resources connected to the local LAN **server**. A single local LAN server or specialized communications server can service multiple remote PC clients. The remote node has the same full capabilities as any local node on the network. The fact that the client PC is remote to the local server is transparent. The data traffic between the remote PC and local LAN server are data packets particular to the network operating system, which both the remote PC and local LAN server have installed.

Remote Control Mode

Remote control mode requires a dedicated local PC to be assigned to each remote PC since applications are stored and executed on the local PC. Shared data and other LAN-attached resources are accessed from the LAN server through the local PC. The remote PC is really nothing more than a simple input/output device. All processing is performed on the local PC. Only keystrokes and screen images are transmitted between the remote PC and the local PC. This setup is really just a remote keyboard and monitor for a local PC. The remote PC *controls* the local PC and all of its LAN attached resources, hence the name, remote control mode.

Figure 22.4 outlines some of the details, features and requirements of these two remote PC modes of operation or logical topologies.

Physical Topologies

Remote connectivity physical topologies refer to the physical arrangement of hardware and media which offers access to the local LAN for the remote user. As Fig. 22.5 illustrates, there are three basic ways in which a remote PC user can gain access to the local LAN resources. (1) a LAN attached PC, (2) communications server, and (3) LAN modem.

© 2006 by Taylor & Francis Group, LLC

Characteristic	Remote Client Operation Mode	Modem Remote Control Operation Mode
Is redirector hardware and/or software required?	Yes	No
Where do the applications actually execute?	On the remote PC	On the local LAN-attached PC or communications server.
What is the nature of the data traffic over the dial-up line?	All interaction between the LAN server and remote PC travels over the dial-up link.	Only keystrokes from the remote PC keyboard and output for the remote PC monitor travel over the dial-up link.
Network interface	Actually using the remote PC's serial port and modem as a substitute NIC. This option provides a very low speed interface.	Uses installed NIC on local LAN attached PC or communication server to interface to the LAN at a very high speed.
Relative performance	Slower than modem remote control operation mode with substantially more traffic, depending whether application programs are stored remotely or on the LAN.	Faster than remote client mode and generates substantially less traffic. Often not appropriate for Windows-based applications
Also called	Remote LAN node. LAN remote control.	

FIGURE 22.4 Remote LAN connectivity: remote PC operations mode.

It is important to understand that the actual implementation of each of these LAN access arrangements may require additional hardware and/or software. They may also be limited in their ability to utilize all LAN attached resources.

22.3 Local Area Networks

LAN Applications

Possible business analysis questions for local area networking solutions are listed in Fig. 22.6. This list of business analysis questions is not meant to be exhaustive or all encompassing. Two important things to remember about any list of business analysis questions are

1. The questions should dig deeper into the required information systems-related business activities.
2. The answers to these questions should provide sufficient insight as to enable the investigation of possible technical solutions.

Next, each of the business analysis questions' categories is explained briefly.

User Issues

User satisfaction is the key to any successful network implementation. To satisfy users, their needs must be first thoroughly understood. Beyond the obvious question of: How many users must the network support? are the more probing questions dealing with specific business activities of individual users. Do users process many short transactions throughout the day? Do users require large file transfers at certain times of day? Are there certain activities which absolutely must be done at certain times of day or within a certain amount of elapsed time? These questions are important in order to establish the amount of network communication required by individual users. Required levels of security should also be addressed. Are payroll files going to be accessed via the network? Who should have access to these files and what security measures will assure authorized access? What is the overall technical ability of the users? Will technical staff need to be hired? Can support be obtained locally from an outside organization?

© 2006 by Taylor & Francis Group, LLC

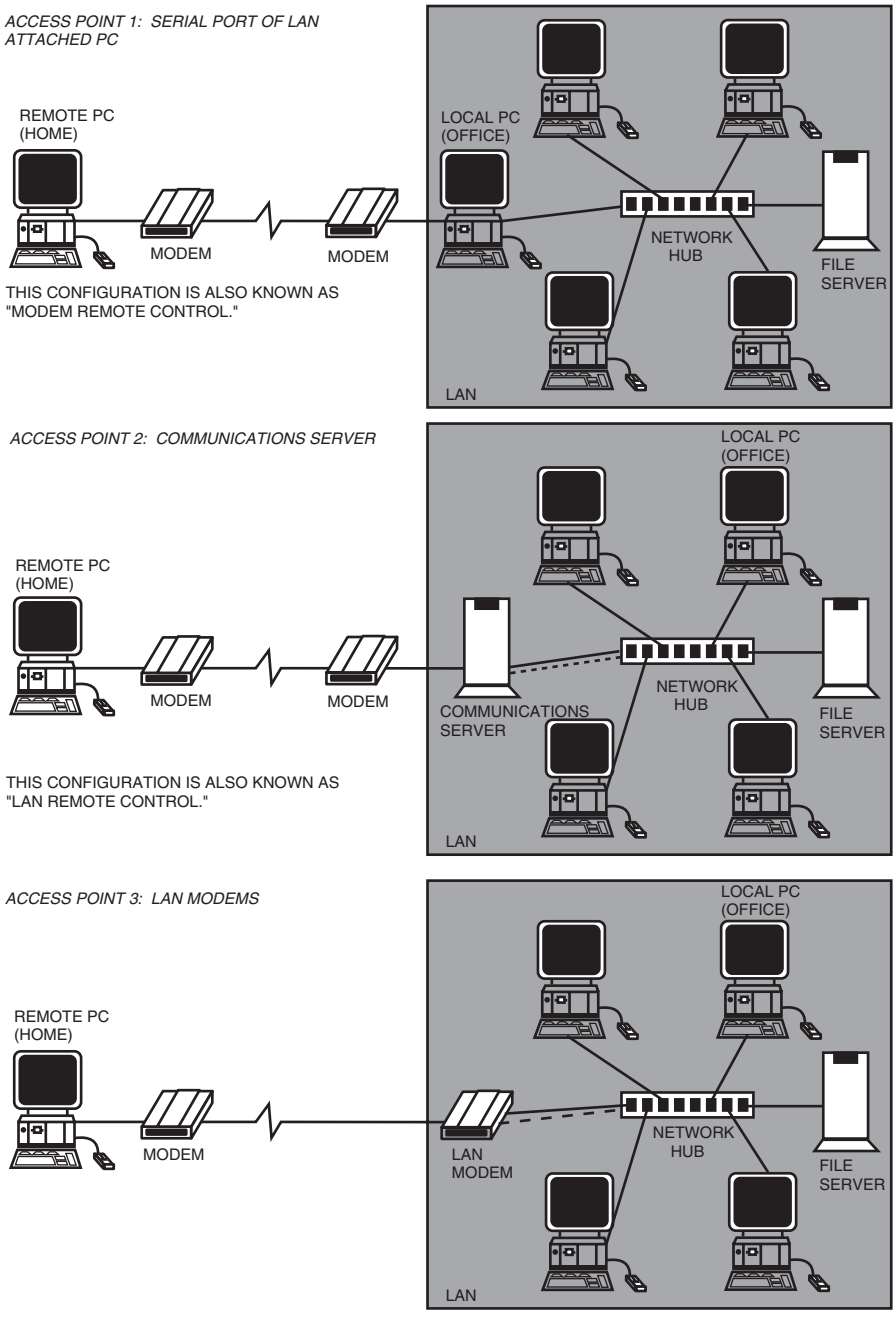


FIGURE 22.5 Remote LAN connectivity: three primary access points.

Local Communication

Remembering that these are business analysis questions and not technical analysis questions, users really cannot be asked how fast their network connections must be. Bits per second or megabits per second have little or no meaning for most users. If users have business activities such as computer aided design/computer aided manufacturing (CAD/CAM) or other three-dimensional modeling or graphics software that will be accessing the network, the network analyst should be aware that these are large consumers of network

© 2006 by Taylor & Francis Group, LLC

	Current	2–3 years	5 years
User issues: How many users? What are their business activities? What is the budgeted cost/user? Comprehensive cost of ownership? What are the security needs? (password protection levels, supervisor privleges) What are the support issues?			
Local communication: Required speed?			
Resource sharing: How many CD-ROMs, printers, modems, and faxes are to be shared? What is the greatest distance from the server to each service?			
File sharing: Is printer/queue management required? How many simultaneous users?			
Application sharing: What is the number and type of required applications? Are e-mail services required?			
Distributed data access: Where will shared data files be stored?			
Lan management/administration: Will training be required to manage the network? How easy is the network to use?			
Extended communication: How many MACs will be part of the network? How many mini/mainframe connections are needed (and what type, IBM, DEC, UNIX based)? Will this be an inter-LAN network? [LAN to LAN concerns. Which NOS? Must other protocols be considered? Are the connections local or remote (long-distance?)]			

FIGURE 22.6 LAN and LAN look alikes: business analysis questions.

bandwidth and should document those information system-related business activities which may be large consumers of networking bandwidth.

Resource Sharing

It is important to identify which resources and how many are to be shared: printers, modems, faxes, and the preferred locations of these shared resources. The required distance between shared resources and users can have a bearing on acceptable technical options.

File Sharing and Application Sharing

Which programs or software packages are users going to need to perform their jobs? Which programs are they currently using? Which new products must be purchased? In many cases, network versions of software packages may cost less than multiple individual licenses of the same software package for individual PCs. The network analyst is really trying at this point to compile a listing of all applications programs which will be shared by users. Not all PC-based software packages are available in network versions and not all PC-based software packages allow simultaneous access by multiple users. Once a complete list of required shared application programs has been completed, it is important to investigate both the availability and

© 2006 by Taylor & Francis Group, LLC

capability of the network versions of these programs in order to assure happy, productive users and the successful attainment of business needs.

Distributed Data Access

Although users cannot be expected to be database analysts, sufficient questions must be asked in order to determine which data is to be shared by whom and where these users are located. The major objective of data distribution analysis is to determine the best location on the network for the storage of various data files. That best location is usually the one closest to the greatest number of the most active users of that data. Some data files that are typically shared, especially in regionalized or multilocation companies, include customer files, employee files, and inventory files. Distributed data access is even more of a concern when the users sharing the data are beyond the reach of a local area network and must share the data via wide area networking solutions. A good starting point for the network analyst might be to ask the question: Has anyone done a comparison of the forms that are used in the various regional and branch offices to determine which data needs to be sent across the network?

Extended Communications

The ability of certain local area networking solutions to communicate beyond the local area network remains a key differentiating factor among local area networking alternatives. Users should be able to articulate connectivity requirements beyond the LAN. The accomplishment of these requirements is the job of the network analyst. Some possible examples of extended communications might include communications to another LAN. If this is the case, the network analyst must investigate all of the technical specifications of this target LAN in order to determine compatibility with the local LAN. The target LAN may be local (within the same building) or remote (across town or around the world). LAN to LAN connection is known as *internetworking* and will be studied in the next section Other examples of extended communications may be the necessity for LAN users to gain access to mainframes, either locally or remotely. Again, users are only asked *what* they need connections to, and *where* those connections must occur, it is the network analyst's job to figure out *how* to make those connections function.

LAN Management and Administration

Another key differentiating factor among LAN alternatives is the level of sophistication required to **manage** and administer the **network**. If the LAN requires a full time, highly trained manager, then that manager's salary should be considered as part of the purchase cost as well as the operational cost of the proposed LAN. Secondly, the users may have requirements for certain management or administration features which must be present. Examples might be user-identification creation or management, or control of access to files or user directories.

Budget Reality

The most comprehensive, well documented, and researched networking proposal is of little value if its costs are beyond the means of the funding organization or business. Initial research into possible networking solutions is often followed by feasibility option reports that outline possible network designs of varying price ranges. Senior management then dictates which options deserve further study based on financial availability. In some cases, senior management may have an approximate project budget in mind which could be shared with network analysts. This acceptable financial range, sometimes expressed as budgeted cost per user, serves as a frame of reference for analysts as technical options are explored. In this sense, budgetary constraints are just another overall, high-level business need or perspective that helps to shape eventual networking proposals.

Anticipated Growth is Key

User needs are not always immediate in nature. These needs can vary dramatically over time. To design networking solutions that will not become obsolete in the near future, it is essential to gain a sense of what the anticipated growth in user demands might be. Imagine the chagrin of the network analyst who must

© 2006 by Taylor & Francis Group, LLC

Network Application	Issues
Network printing	Printers shared; printer requests, buffered and spooled; 2nd generation network printers faster/smarter. Queue/device management, multiple print formats.
Network backup	Match hardware and software to your needs and network. Check out restore features. Scheduling unattended backups. Generation of log reports. Hardware dependencies and capacities.
Network security	Virus control. Additional User ID/password protection. Encryption. User authentication. Automatic logout of inactive terminals.
Network management	Network monitors. Diagnostic and analysis tools, which may be seperated depending on cable and network type. Administration concerns. Software often in NOS for simple LANs.
Network resource optimization Configuration mgmt. Inventory mgmt.	Hardware and Software inventory database. Reporting and query capabilities. Planning and network configuration tools. Monitoring and network alarm capabilities.
Remote access-control software	Included in many fully integrated LANs.
Connectivity/gateway software MAC integration software	
Groupware	Workflow automation. Interactive work document review. Information sharing. Group schedules. Enhanced e-mail.

FIGURE 22.7 LAN and LAN look alikes: network applications analysis.

explain to management that the network which was installed last year cannot be expanded and must be replaced due to unanticipated growth of network demand. One method of gaining the necessary insight into future networking requirements, illustrated in Fig. 22.6, is to ask users the same set of business analysis questions with projected time horizons of 2–3 years and 5 years. Incredible as it may seem, 5 years is about the maximum projected lifetime for a given network architecture or design. Of course, there are exceptions. End users may not have the information or knowledge necessary to make these projections. Management can be very helpful in the area of projected growth and informational needs, especially if the company has engaged in any sort of formalized strategic planning methodology.

Network Applications: What Can a LAN Do for You?

Beyond merely being able to share the same application software packages (spreadsheets, word processing, databases), which ran on individuals PCs before they were networked together over a LAN, networking PCs together provides some unique opportunities to run additional networked applications, which can significantly increase worker productivity and/or decrease costs.

Figure 22.7 summarizes the attributes and issues of some of the most popular uses of a LAN. It should be noted that the uses, features, and issues described next apply only to the listed applications functioning on a local area network. Many of these same functions become much more complicated when running across multiple LANs (internetworking) or over long distance WANs. A short description of each of these LAN applications follows

Network Printing

Network printing continues to evolve as user demands change and technology changes to meet those demands.

On a typical LAN, a networked PC would send a request for printing out onto the network through a network interface card. The networked request for printing services would be accepted by a device in

© 2006 by Taylor & Francis Group, LLC

charge of organizing the print requests for a networked printer. Depending on the LAN implementation configuration, that device may be a PC with attached printer, a specialized print server with attached printers, or a directly networked printer. Some type of software must manage all of this requesting, spooling, buffering, queuing and printing. The required software may be part of an overall network operating system or may be specifically written for only network printer management.

Network Backup

Backing up data and application files on a network is essential to overall network security and the ability to recover from the inevitable data-destroying disaster. Although the process and overall components are relatively simple, the implementation can be anything but simple. Basically there are only two components to a network backup system: (1) the software, which manages the backup, and (2) the hardware device, which captures the backed-up files.

Some network backup software and hardware work with only certain network operating systems. Other network backup software will work with only the hardware device with which it is sold. The interaction between hardware devices and software such as operating systems or network operating systems is often controlled by specialized software programs (drivers). It is essential to make sure that the necessary drivers are supplied by either the tape backup device vendor or software vendor in order to ensure the operability of the tape backup device. Hardware devices may be various types of tape subsystems, or optical drives. Key differences among hardware devices include

1. How much? What is the storage capacity of the backup device?
2. How fast? How quickly can data be transferred from a PC or server to the backup device? This attribute is important if you are backing up large capacity disk drives.
3. How compressed? Can data be stored on the backup device in compressed form? If so, it may save significant room on the backup media.

Remember, backup is not necessarily a one way process. Restoration of backed up files and the ease with which that restoration can be accomplished is a major purchasing consideration. Being able to schedule unattended backups or restorals as well as the ability to spool or print log reports of backup/restoral activity are also important to **network managers**.

Network Management

The overall task of network management is usually broken down into at least three distinct areas of operation. First, networks must be *monitored*. Therefore, one of the foremost jobs of network management software is to monitor the LAN to detect any irregular activity such as malfunctioning network adapter cards, or an unusually high data transmission rate monopolizing the available network bandwidth to be shared among all attached workstations. Sophisticated LAN monitoring programs can display maps of the network on graphics terminals. Operators can zoom in on a particular node or workstation for more information or performance statistics. Some monitoring programs also have the ability to compare current network activity to preset acceptable parameters and to set off alarms on network monitor terminals, perhaps by turning a node symbol bright red on the screen, when activity goes beyond acceptable limits. Monitoring software is also written specifically for monitoring file servers on a LAN. Besides monitoring server performance and setting off alarms, some monitoring packages have the ability to dial and set off pagers belonging to network managers who may be away from the network console.

Once a problem has been monitored and identified, it must be *analyzed* and *diagnosed*. This is the second major task of network management software. Diagnosis is often done by a class of devices known as protocol analyzers or by the more common name, sniffers. These devices are attached to the LAN and watch, measure, and in some cases, record every bit of data that passes their way. By using multiple sniffers at various points on a LAN, otherwise known as distributed sniffers, bottlenecks can be identified and performance degradation factors can be pinpointed. LAN testing devices must be able to test and isolate the three major segments of any LAN: (1) the wire or cable of the network, (2) the network adapter cards, which interface between the cable and the workstation, and (3) the workstation or PC that generates the network activity.

© 2006 by Taylor & Francis Group, LLC

Having diagnosed the cause of the networking problem, corrective action must be taken against that problem. Perhaps a misbehaving network interface card must be disabled or an application on a workstation that is monopolizing network resources must be logged out. The power to do these things is sometimes called network *administration* or management. In this case, the term administration is preferable in order to provide a contrast to the previous more general use of the term network management. Most often, the required network management software to manage LANs is included in the network operating system itself.

LAN monitoring software and other specialized network management functions are available as an add-on product for most network operating systems. When these add-on products are manufactured by a company other than the original network operating system vendor, they are known as third-party products. These third-party enhancements are often of high quality but should be purchased with caution. Compatibility with associated software or future releases of the network operating system are not necessarily guaranteed.

Network Security

In addition to the typical security features, such as password protection and directory access control supplied with most network operating systems, more sophisticated network security software/hardware is available for LANs. For instance, security software may be added to workstations and or servers which will

1. Require user identification and valid password to be entered before the PC can be booted.
2. Encrypt important data or application files to prevent tampering.
3. Automatically logout inactive terminals to prevent unauthorized access to system resources.
4. Allow certain users to run only certain applications.
5. Require verification of user authenticity by security verifiers.

Another area of network security that is receiving a lot of attention is that of *viruses*. Virus control software is sometimes included in network security packages. Virus control is really a three-step process, implying that effective virus control software should address at least the following three areas:

1. Virus protection: User access to systems is sufficiently controlled as to prevent an unauthorized user from infecting the LAN.
2. Virus detection: Sophisticated software to find viruses regardless of how cleverly they may be disguised.
3. Virus eradication: Sometimes called antibiotic programs, this software eliminates all traces of the virus.

Groupware

Groupware is the name of a category of software that seeks to take advantage of the fact that workers are networked together electronically in order to maximize worker productivity. Groupware is a general term that describes all or some of the following software categories: workflow automation, interactive work, group scheduling, document review, information sharing, electronic whiteboard, and enhanced electronic mail.

Local Area Network Architectures

The choice of a particular network architecture will have a definite bearing on the choice of network adapter cards and less of an impact on the choice of media or network operating system. For instance, an ethernet network architecture requires ethernet adapter cards. As will soon be seen, it is the adapter card which holds the key, or media access control (MAC) layer protocol, which determines whether a network is ethernet, token ring, **fiber distributed data interface (FDDI)** or any other network architecture. Ethernet runs over thick or thin coaxial cable, shielded or unshielded twisted pair, fiber or wireless—clearly a wide choice of media options.

© 2006 by Taylor & Francis Group, LLC

Ethernet

Ethernet, adhering to the IEEE 802.3 standard, is a **carrier sense multiple access with collision detection-(CSMA/CD-)** based network architecture traditionally installed in a bus configuration, but most often installed in a hub-based star physical topology. Every device, most often network adapter cards, attached to an ethernet network has a unique hardware address assigned to it at time of manufacture. As new devices are added to an ethernet network, their addresses become new possible destinations for all other attached ethernet devices.

The media access layer protocol elements of ethernet form data packets for transmission over the shared media according to a standardized format. This ethernet packet format is nearly equivalent to the IEEE 802.3 standard, and the two terms are often used interchangeably.

The potential for collisions and retransmission exists on an ethernet network thanks to its CSMA/CD access methodology. In some cases, ethernet networks with between 100 and 200 users barely use the capacity of the network. However, the nature of the data transmitted is the key to determining potential network capacity problems. Character-based transmissions, such as typical data entry, in which a few characters at a time are typed and sent over the network are much less likely to cause network capacity problems than the transfer of graphical user interface (GUI) screen oriented transmission such as Windows-based applications. CAD/CAM images are even more bandwidth intensive. Simultaneous requests for full screen Windows-based transfers by 30 or more workstations can cause collision and network capacity problems on an ethernet network. As with any data communication problem, there are always solutions or workarounds to these problems. The point in relaying these examples is to provide some assurance that although ethernet is not unlimited in its network capacity, in most cases, it provides more than enough bandwidth.

Token Ring

IBM's token ring network architecture, adhering to the IEEE 802.5 standard, utilizes a star physical topology, sequential message delivery, and a token passing access methodology scheme. Since the sequential logical topology is equivalent to passing messages from neighbor to neighbor around a ring, the token ring network architecture is sometimes referred to as: *logical ring, physical star*. The token ring's use of the token passing access methodology furnishes one of the key positive attributes of this network architecture. The guarantee of no data collisions with assured data delivery afforded by the token passing access methodology is a key selling point in some environments where immediate, guaranteed delivery is essential.

FDDI

As far as trends in network architecture go, as more and more users are attached to LANS, the demand for overall network bandwidth increases. LANs are increasing both in size and overall complexity. Internetworking of LANs of various protocols via bridges and routers means more overall LAN traffic. Network applications are driving the demand for increased bandwidth as well. The concepts of distributed computing, data distribution, and client/server computing all rely on a network architecture foundation of high bandwidth and high reliability. Imaging, multimedia, and data/voice integration all require high amounts of bandwidth in order to transport and display these various data formats in real time. In other words, if full-motion video is to be transported across the LAN as part of a multimedia program, there should be sufficient bandwidth available on that LAN for the video to run at full speed and not in slow motion. Likewise, digitized voice transmission should sound normal when transported across a LAN of sufficient bandwidth.

FDDI supplies not only a great deal of bandwidth, but also a high degree of reliability and security while adhering to standards-based protocols. FDDI's reliability comes not only from the fiber itself which, as we know, is immune to both electromagnetic interference (EMI) and radio frequency interference (RFI). An additional degree of reliability is achieved through the design of the physical topology of FDDI.

FDDI's physical topology comprises not one, but two, separate rings around which data moves simultaneously in opposite directions. One ring is the primary data ring while the other is a secondary or backup

© 2006 by Taylor & Francis Group, LLC

data ring to be used only in the case of the failure of the primary ring. Whereas both rings are attached to a single hub or concentrator, a single point of failure remains in the hub while achieving redundancy in the network media. In addition to speed and reliability, distance is another key feature of an FDDI LAN.

Another positive attribute of FDDI is its ability to interoperate easily with ethernet networks. In this way, a business does not have to scrap its entire existing network in order to upgrade a piece of it to FDDI. An FDDI to ethernet bridge is the specific technology employed in such a setup.

The uses of the FDDI network architecture typically fall into three categories:

1. Campus backbone: Not necessarily implying a college campus, this implementation is used for connecting LANs located throughout a series of closely situated buildings.
2. High bandwidth workgroups: The second application category is when the FDDI LAN is used as a truly local area network, connecting a few PCs or workstations which require high bandwidth communication with each other. Multimedia workstations, engineering workstations, or CAD/CAM workstations are all good examples of high bandwidth workstations.
3. High bandwidth subworkgroup connections: In some cases, only two or three devices, perhaps three servers, have high bandwidth requirements. As distributing computing and data distribution increase, an increasing demand for high-speed server to server data transfer has been seen.

Wireless LANs

Many of the previously mentioned network architectures function over more than one type of media. Another media option is wireless transmission (which is really the absence of any media) for local area networks. There are currently three popular wireless transmission technologies in the local area network technology area. They are microwave transmission, spread spectrum transmission, and infrared transmission.

These are all just radio transmissions at varying frequencies.

Wireless LAN Applications

A primary application of **wireless LANs** optimizes the ease of access of the wireless technology. Portable or notebook PCs equipped with their own wireless LAN adapters can create an instant LAN connection merely by getting within range of a server-based wireless LAN adapter or wireless hub. In this way, a student or employee can sit down anywhere and log into a LAN as long as the user is within range of the wireless hub and has the proper wireless adapter installed in the portable PC.

Meeting rooms can be equipped with wireless hubs to allow spontaneous workgroups to log into network resources without running cables all over the meeting room. Similarly, by quickly installing wireless hubs and portable PCs with wireless adapters, temporary expansion needs or emergency/disaster recovery situations can be handled quickly and with relative ease. No rerunning of wires or finding the proper cross-connects in the wiring closet are necessary.

Finally, wireless LAN technology allows entire LANs to be preconfigured at a central site and shipped ready to run to remote sites. The nontechnical users at the remote site literally just have to plug the power cords into the electrical outlets and they have an instant LAN. For companies with a great number of remote sites and limited technical staff, such a technology is ideal. No preinstallation site visits are necessary. Also avoided are costs and supervision of building wiring jobs and troubleshooting building wiring problems during and after installation.

Local Area Network Hardware

Servers

A server's job is to manage the sharing of networked resources among client PCs. Depending on the number of client PCs and the extent of the shared resources, it may be necessary to have multiple servers and/or to specialize some servers as to the type of resources that they manage. If servers are to be specialized, then

© 2006 by Taylor & Francis Group, LLC

shared resources should be grouped in some logical fashion so as to optimize the server performance for managing the sharing of a particular type of resource. A list of potentially shared network resources would probably include: files, application programs, databases, printers, access to other LANs (local), access to other LANs (remote), access to information services, and access to the LAN from remote PCs.

Hubs/Multistation Access Units (MAUs)

The heart of the star physical topology, employed by both ethernet and token ring, is the wiring center, alternatively known as a hub, a concentrator, a repeater or a multistation access unit (MAU).

Repeaters

A repeater, as its name would imply, merely repeats each bit of digital data that it receives. This repeating action actually cleans up the digital signals by retiming and regenerating them before passing this repeated data from one attached device or LAN segment to the next.

Hubs

The terms hub and *concentrator* or *intelligent concentrator* are often used interchangeably. Distinctions can be made, however, between these two broad classes of wiring centers, although there is nothing to stop manufacturers from using the terms as they wish. A hub is often the term reserved for describing a stand-alone device with a fixed number of ports, which offers features beyond that of a simple repeater. The type of media connections and network architecture offered by the hub is determined at the time of manufacture as well. For example, a 10BaseT ethernet hub will offer a fixed number of RJ-45 twisted pair connections for an ethernet network. Additional types of media or network architectures are not usually supported.

MAUs

A MAU is IBM's name for a token ring hub. A MAU would be manufactured with a fixed number of ports and connections for unshielded or shielded twisted pair. IBM uses special connectors for token ring over shielded twisted pair (STP) connections to a MAU. MAUs offer varying degrees of management capability.

Concentrators

The term concentrator or intelligent concentrator (or smart hub) is often reserved for a device characterized by its flexibility and expandability. A concentrator starts with a fairly empty, boxlike device often called a chassis. This chassis contains one or more power supplies and a builtin network backbone. This backbone might be ethernet, token ring, FDDI, or some combination of these. Into this backplane, individual cards or modules are inserted.

For instance, an 8- or 16-port twisted pair ethernet module could be purchased and slid into place in the concentrator chassis. A network management module supporting the SNMP (simple network management protocol) network management protocol could then be purchased and slid into the chassis next to the previously installed 10BaseT Port module. In this mix and match scenario, additional cards could be added for connection of PCs with token ring adapters, PCs, or workstations with FDDI adapters, or dumb asynchronous terminals.

This network in a box is now ready for workstations to be hooked up to it through twisted pair connections to the media interfaces on the network interface cards of the PCs or workstations. Remember that ethernet can run over UTP (unshielded twisted pair), STP, thick and thin coaxial as well as fiber.

Additional modules available for some, but not all, concentrators may allow data traffic from this network in box to travel to other local LANs via bridge or router add-on modules. (Bridges and routers will be discussed in the next section on internetworking.) These combination concentrators are sometimes called *internetworking hubs*. Communication to remote LANs or workstations may be available through the addition of other specialized cards, or modules, designed to provide access to wide area network services purchased from common carriers.

© 2006 by Taylor & Francis Group, LLC

Switching Hubs

The network in a box or backbone in a box offered by concentrators and hubs shrinks the length of the network backbone but does not change the architectural characteristics of a particular network backbone. For instance, in an ethernet concentrator, multiple workstations may access the built in ethernet backbone via a variety of media, but the basic rules of ethernet, such as CSMA/CD access methodology still control performance on this ethernet in a box. Only one workstation at a time can broadcast its message onto the shared backbone.

A *switching hub* seeks to overcome this one at a time broadcast scheme, which can potentially lead to data collisions, retransmissions, and reduced throughput between high-bandwidth demanding devices such as engineering workstations or server-to-server communications.

The ethernet switch is actually able to create connections, or switch, between any two attached ethernet devices on a packet by packet basis. The one-at-time broadcast limitation previously associated with ethernet is overcome with an ethernet switch.

Wiring Centers Technology Analysis

Some of the major technical features to be used for comparative analysis are listed in Fig. 22.8. Before purchasing a hub of any type, consider the implications of these various possible features. To summarize, the following major criteria should be thoroughly considered before a hub or concentrator purchase: (1) expandability (2) supported network architectures (3) supported media types (4) extended communications capabilities, that is, terminal support, internetworking options, and wide area networking options, (5) hub/concentrator management capabilities and (6) reliability features.

Network Interface Cards

Network adapter cards, also known as network interface cards are the physical link between a client or server PC and the shared media of the network. Providing this interface between the network and the PC or workstation requires that the network adapter card have the ability to adhere to the access methodology (CSMA/CD or token passing) of the network architecture to which it is attached. These software rules, implemented by the network adapter card which control the access to the shared network media, are known as *media access control* (MAC) protocols and are represented on the MAC sublayer of the data link layer (layer 2) of the OSI 7-layer reference model.

Since these are MAC layer interface cards and are, therefore, the keepers of the MAC layer interface protocol, it is fair to say that it is the adapter cards themselves that determine network architecture and its constituent protocols more than any other component. Take an ethernet adapter card out of the expansion slot of a PC and replace it with a token ring adapter card and you have a token ring workstation. In this same scenario, the media may not even need to be changed since ethernet, token ring, and FDDI/CDDI often work over the same media.

Role of Adapter Card Drivers

Assuring that the purchased adapter card interfaces successfully to the bus of the CPU as well as the chosen media of the network architecture will as sure hardware connectivity. Full interoperability, however, depends on the chosen network adapter card being able to communicate successfully with the network operating system and operating system of the PC into which it is installed.

22.4 Internetworking

Applications: What Can Internetworking Do for You?

It is nearly inevitable that sooner or later nearly any organization will need to share information across multiple information platforms or architectures. This sharing may be between LANs that may differ in network architecture or network operating systems. Information systems that combine multiple computing platforms or a variety of network architectures and network operating systems are often referred to as

© 2006 by Taylor & Francis Group, LLC

Characteristic/Feature	Options/Implications
Expandability	Options: ports, LANs, slots. Remember, most stand-alone hubs are not expandable, although they may be cascaded. Concentrators may vary in their overall expandability (number of open slots,) or in their ability to add additional ports. Several concentrators allow only one LAN backbone module. Have you anticipated growth in 2- and 5-year horizons?
Network Architecture	Options: ethernet, token ring, Arcnet, FDDI, Appletalk. Some concentrators allow only one type of LAN module, with ethernet being the most widely supported type of LAN module. Before purchasing concentrator, make sure all of the necessary LAN types are supported.
Media	Options: Unshielded twisted pair, shielded twisted pair, thin coax, thick coax, fiber. Remember that each of these different media will connect to the hub with a different type of interface. Make sure that the port modules offer the correct interfaces for attachment of your media; also, remember that a NIC is on the other end of the attached media. Is this concentrator/hub compatible with your network adapter cards?
Extended Communications:	
Macintosh support	Can an Apple Macintosh be linked to the hub?
Terminal support	Are modules available which suuport direct connection of dumb asynchronous terminals? What is the interface (i.e., DB-25, RJ-45)?
Internetworking	Options: Internetworking to: ethernet, token ring, Arcnet, FDDI, Appletalk. Internetworking solutions are unique, dependent on the two LANS to be connected. In other words, unless a concentrator offers a module which specifically internetworks ethernet module traffic to token ring LAN, this communication can not take place within the concentrator.
Wide Area Networking	Options: Interfaces to different carrier services—DDS, DS-0, T-1, X.25, frame relay, SMDS, ATM.
Management	Options: Protocols: SNMP, CMIP, CMOT Options: What level of management is offered? 1. Can individual ports be managed? 2. Is monitoring software included? 3. Can wide area as well as local area link performance be analyzed/monitored through the concentrator? 4. What level of security in general or regarding management functions specifically is offered? 5. Can ports be remotely enabled/disabled? 6. Can the hub/concentrator be controlled from any attached workstation? Via modem? 7. Can management/monitoring functions be included through a graphical user interface? 8. How are alarm thresholds set? 9. How are faults managed? 10. Can port access be limited by day and/or time? 11. What operating systems can the management software run on? DOS, OS/2, Windows NT, UNIX, APPLE? 12. Can a map of the network be displayed?
Reliability	Options: 1. Is integrated UPS included? 2. Are there multiple, redundant power supplies? 3. Can individual modules be swapped out "hot", without powering down the entire hub

FIGURE 22.8 Wiring center technology analysis grid.

enterprise computing environments. Underlying this enterprise computing environment is an *enterprise network* or *internetwork*. The key to successful *internetworking* is that to the end user sitting at a LAN-attached workstation, the connectivity to enterprise computing resources should be completely transparent.

In other words, an end user should not need to know the physical location of a database server or disk drive to which they need access. All the user needs to know is a node name or drive indicator letter and the fact that when these node name or drive letters are entered, data is properly accessed from wherever it may be physically located. That physical location may be across the room or across the country.

© 2006 by Taylor & Francis Group, LLC

As information has become increasingly recognized as a corporate asset to be leveraged to competitive advantage, the delivery of that information in the right form, to the right person at the right place and time has become the goal of information systems and the role of internetworking. Mergers, acquisitions, and enterprise partnerships have accelerated the need to share information seamlessly across geographic or physical LAN boundaries. Intelligent inter-LAN devices perform the task of keeping track of what network attached resources are located and where and how to get a packet of data from one LAN to another. This task is further complicated by the fact that LANS can differ in any of the following categories: (1) network architecture, (2) media, (3) network operating system, and (4) operating system.

This collection of differences is defined by rules or *protocols* that define how certain clearly defined aspects of network communications are to take place. When LANs that need to share information operate according to different protocols, an inter-LAN device that has *protocol conversion* capabilities must be employed. In cases such as this, the inter-LAN device would be transmitting data across logical, as opposed to physical or geographic, boundaries. Among the inter-LAN devices which will be explored in this section are repeaters, bridges, routers, and gateways. Although the differences in functionality of these devices will be distinguished from a technical standpoint, there is no guarantee that manufacturers will follow this differentiation. Functional analysis of data communications devices is the best way to assure that inter-LAN devices will meet internetworking connectivity needs. Internetworking analysis and design is a highly complex and confusing area of study. The purpose of this section is to familiarize the reader with internetworking technology and applications and to provide a list of resources for further study.

The OSI Model and Internetworking Devices

The OSI model was first introduced in Sec. 22.1 as a common framework or reference model within which protocols and networking systems could be developed with some assurance of interoperability. In this section, the OSI model provides a most effective framework in which to distinguish between the operational characteristics of the previously mentioned internetworking devices. Figure 22.9 depicts the

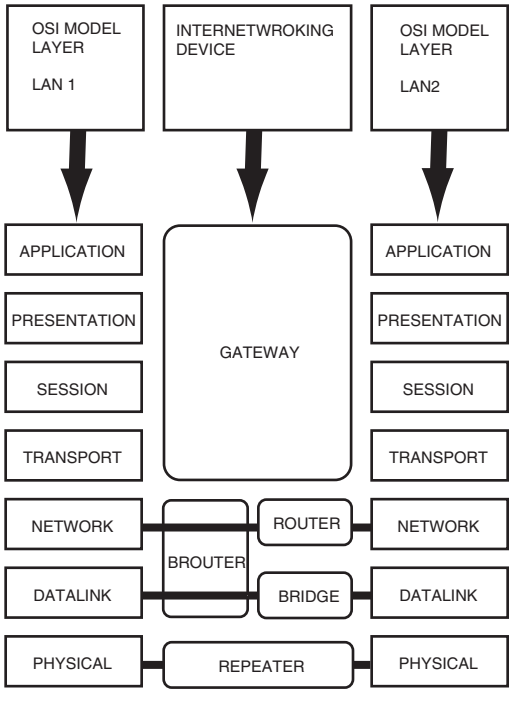


FIGURE 22.9 Relationship between the OSI model and internetworking devices.

© 2006 by Taylor & Francis Group, LLC

relationship between each type of internetworking device and its related OSI layer. Each device as well as the related OSI model layer will be explained in detail.

Internetworking Technology

Repeaters: Layer 1---The Physical Layer

Remember that all data traffic on a LAN is in a digital format of discrete voltages of discrete duration traveling over one type of physical media or another. Given this, a repeater's job is fairly simple to understand:

1. Repeat the digital signal by regenerating and retiming the incoming signal.
2. Pass all signals between all attached segments.
3. Do not read destination addresses of data packets.
4. Allow for the connection of different types of media.
5. Effectively extend overall LAN distance by repeating signals between LAN segments.

A repeater is a nondiscriminatory internetworking device. It does not discriminate between data packets. Every signal which comes into one side of a repeater gets regenerated and sent out the other side of the repeater. Repeaters are available for both ethernet and token ring network architectures for a wide variety of media types. A repeater is a physical layer device concerned with physical layer signaling protocols relating to signal voltage levels and timing. It cannot distinguish between upper layer protocols such as between ethernet vs. token ring frames (layer 2 , data link protocols). Therefore, repeaters must be specifically manufactured for either ethernet or token ring network architectures. The primary reasons for employing a repeater are (1) increase the overall length of the network media by repeating signals across multiple LAN segments, (2) isolate key network resources onto different LAN segments, and (3) some repeaters also allow network segments of the same network architecture but different media (layer 1—physical) types to be interconnected.

Bridges: Layer 2---The Data Link Layer

Bridge Functionality

The primary reasons for employing bridges are (1) network traffic on a LAN segment has increased to the point where performance is suffering, and (2) access from the departmental LAN to the corporate LAN backbone needs to be controlled so that local LAN data is not unnecessarily causing congestion problems on the corporate backbone network.

By dividing users across multiple LAN segments connected by a bridge, a substantial reduction in LAN traffic on each segment can be achieved provided the division of users is done in a logical manner. Users should be divided according to job function, need to communicate with each other, and need to access data stored on particular servers. The rule of thumb for segmenting users is that 80% of LAN traffic should remain within the LAN segment and only about 20% should cross the bridge to adjacent LAN segments.

Controlling access to the corporate backbone via a bridge can ensure the viability of enterprise communications by only allowing essential network communication onto the corporate backbone. Servers and other internetworking devices can be connected directly to the corporate backbone while all user's workstations are connected to LAN segments isolated from the corporate backbone by bridges.

When users on one LAN need occasional access to data or resources from another LAN, an internetworking device, which is more sophisticated and discriminating than a repeater, is required. From a comparative standpoint on the functionality of bridges vs repeaters, one could say that bridges are more discriminating. Rather than merely transferring *all* data between LANs or LAN segments like a repeater, a bridge reads the *destination address* of each data frame on a LAN, decides whether the destination is local or remote (on the other side of the bridge), and only allows those data frames with nonlocal destination addresses to cross the bridge to the remote LAN.

How does the bridge know whether a destination is local or not? Data-link protocols such as ethernet contain *source addresses* as well as destination addresses within the predefined ethernet frame layout. A bridge also checks the source address of each frame it receives and adds that source address to a table of

© 2006 by Taylor & Francis Group, LLC

Bridge Type	LAN Segment Linked
Transparent	• Ethernet to ethernet • Nonsource routing token ring to nonsource routing token ring
Translating	• Ethernet to token ring and vice versa
Encapsulating	• Ethernet to FDDI and vice versa
Source routing	• Source routing token ring to source routing token ring
Source routing transparent	• Source routing token ring to ethernet and vice versa
Adaptive source routing transparent	• Ethernet to ethernet • Source routing token ring to source routing token ring • Source routing token ring to ethernet and vice versa

FIGURE 22.10 Bridges link LAN segments.

known local nodes. After each destination address is read, it is compared with the contents of the known local nodes table in order to determine whether the frame should be allowed to cross the bridge or not (whether the destination is local or not). Bridges are sometimes known as a forward-if-not-local device.

This reading, processing, and discriminating indicates a higher level of sophistication of the bridge, afforded by installed software.

Bridge Categorization

Bridges come in many varieties. Physically, bridges may be cards that can be plugged into an expansion slot of a PC, or they may be standalone devices. Although it is known that the bridge will do the internetwork processing between two LANs, the exact nature of that processing, as well as the bridge's input and output interfaces, will be determined by the characteristics of the two LANs that the bridge is internetworking. In determining the attributes of the input and output bridge one must consider the following issues: MAC sublayer protocol, speed of LANs, local or remote, and wide area network services and media.

1. MAC Sub-Layer Protocol: Depending on the MAC sublayer or network architecture of the LANs to be bridged, any of the following types of bridges may be required: transparent bridges, translating bridges, encapsulating bridges, source routing bridges, source routing transparent bridges, and adaptive source routing transparent bridges. First and foremost, are the two LANs which are to be bridged ethernet or token ring? Bridges that connect LANs of similar data link format are known as *transparent bridges*. A special type of bridge that includes a *format converter* can bridge between ethernet and token ring. These special bridges may also be called *multiprotocol bridges* or *translating bridges*. A third type of bridge, somewhat like a translating bridge, is used to bridge between ethernet and FDDI networks. Unlike the translating bridge, which must actually manipulate the data-link layer message before repackaging it, the *encapsulating bridge* merely takes the entire ethernet data link layer message and stuffs it in an envelope (data frame) that conforms to the FDDI data-link layer protocol. *Source routing bridges* are specifically designed for connecting token ring LANs. Bridges that can support links between source routing token ring LANs and nonsource routing LANs, such as ethernet, are known as *source routing transparent bridges*. Finally, bridges that can link transparent bridged ethernet LAN segments to each other, source routing token ring LAN segments to each other, or any combination of the two are known as *adaptive source routing transparent bridges*. Figure 22.10 outlines these various bridge possibilities.
2. Speed of LANs: The speeds of the input and output LANs must be known in order to determine what speed conversion, if any, must be performed by our bridge.

© 2006 by Taylor & Francis Group, LLC

3. Local or Remote: Having determined the MAC layer protocol and speed of the LANs, their geographic proximity to one another must be taken into consideration. If the two LANs are not in close enough proximity to link via traditional LAN media such as UTP, coax, or fiber, the bridge must be equipped with an interface appropriate for linking to wide area carrier services.

Bridge Performance

Bridge performance is generally measured by two criteria:

1. *Filtering rate:* Measured in packets per second or frames per second. When a bridge reads the destination address on an ethernet frame or token ring packet and decides whether or not that packet should be allowed access to the internetwork through the bridge, that process is known as *filtering*.
2. *Forwarding rate:* Also measured in packets per second or frames per second. Having decided whether or not to grant a packet access to the internetwork in the filtering process, the bridge now must perform a separate operation of forwarding the packet onto the internetwork media whether local or remote.

Bridges, Protocols, and the OSI Model

Bridges read the destination addresses within data frames of a predefined structure or protocol. In other words, ethernet and token ring network architectures define a bit-by-bit protocol for formation of data frames. The bridge can rely on this protocol and, therefore, knows just where to look within the ethernet data frames to find the bits, which represent the destination addresses. In terms of the OSI model, ethernet and token ring are considered MAC sublayer protocols. The MAC sublayer is one of two sublayers of OSI model layer 2—the data link layer. The other data link sublayer is known as the *logical link control* sublayer. Because the protocols that a bridge reads and processes are located on the MAC sublayer, bridges are sometimes referred to as MAC layer bridges.

Embedded within the data field of the ethernet frame are all of the higher OSI layer protocols. These higher layer protocols can vary independently of the data-link layer ethernet protocol. In other words, the data-link layer protocols such as ethernet and token ring are *network architectures*, whereas the network layer protocols could be from any one of a number of different *network operating systems*. Bridges only pay attention to network architecture (MAC sublayer) protocols or formats. They completely ignore upper level protocols.

Most network operating systems actually consist of stacks of protocols. In some cases, this *protocol stack* may consist of a separate protocol for each of layers 3–7. Each protocol of a network operating system performs a different networking related function corresponding to the generalized functional definition for the corresponding layer of the OSI model. As an example, the network layer protocol for TCP/IP suite of protocols is known as *internet protocol* (IP).

Routers: The Network Layer Processors

The delivery of data packets to destination addresses across multiple LANs, and perhaps over wide area network links, is the responsibility of a class of internetworking devices known as *routers*. Routers are primarily employed for the following reasons:

1. To build large hierarchical networks. Routers are used to create the backbone network itself.
2. To take part in or gain access to a larger hierarchical network such as the Internet.

Router Functionality

Although they both examine and forward data packets, routers and bridges differ significantly in two key functional areas. First, although a bridge reads the destination address of every data packet on the LAN to which it is attached, a router only examines those data packets that are specifically addressed to it. Second, rather than just merely allowing the data packet access to the internetwork in a manner similar to a bridge, a router is more cautious as well as more helpful. Before indiscriminately forwarding a data packet, a router first confirms the existence of the destination address as well as the latest information on available

© 2006 by Taylor & Francis Group, LLC

TABLE 22.1 Network and Data-Link Protocols

Network Layer Protocol	Network Operating System or Protocol Stack Name
IPX	NetWare
IP	TCP/IP
VIP	Vines
AFP	Appletalk
XNS	3Com
OSI	Open Systems
Other Protocols	
LAT	Digital DecNet
SNA/SDLC	IBM SNA
Netbios	DOS-based LANs

network paths to reach that destination. Next, based on the latest traffic conditions, the router chooses the best path for the data packet to reach its destination and sends the data packet on its way. The word best is a relative term, controlled by a number of different protocols, which will be examined shortly.

The router itself is a destination address, available to receive, examine, and forward data packets from anywhere on any network to which it is either directly or indirectly internetworked. The destination address on an ethernet or token ring packet must be the address of the router that will handle further internetwork forwarding. Thus, a router is addressed in the data-link layer destination address field. The router then discards this MAC sublayer envelope, which contained its address, and proceeds to read the contents of the data field of the ethernet or token ring frame.

Just as in the case of the data-link layer protocols, network layer protocols dictate a bit-by-bit data frame structure that the router understands. What looked like just data and was ignored by the data-link layer internetworking device (the bridge) is unwrapped by the router and examined thoroughly in order to determine further processing. After reading the network layer destination address and the protocol of the network layer data, the router consults its *routing tables* in order to determine the best path on which to forward this data packet. Having found the best path, the router has the ability to repackage the data packet as required for the delivery route (best path) it has chosen.

As an example, if a packet switched data network was chosen as the wide area link for delivery, then the local router would encapsulate the data packet in compliant envelope. On the other hand, if the best path was over a local ethernet connection, the local router would put the data packet back into a fresh ethernet envelope and send it on its way.

Unlike the bridge, which merely allows access to the internetwork (forward-if-not-local logic), the router specifically addresses the data packet to a distant router. Before a router actually releases a data packet onto the internetwork, however, it confirms the existence of the destination address to which this data packet is bound. Only once the router is satisfied with both the viability of the destination address, as well as with the quality of the intended path, will it release the carefully packaged data packet. This meticulous processing activity on the part of the router is known as *forward-if-proven-remote* logic.

Determination of Best Path

The best path can take into account variables such as

1. Number of intermediate hops. That is, how many other routers will the packet have to be processed by before it reaches its final destination? Every router takes time to process the data packet. Therefore, the fewer the routers, the faster the overall delivery.
2. The speed or condition of the communications circuits. Routers can dynamically maintain their routing tables keeping up to the minute information on network traffic conditions.
3. The protocol of the network operating system, for instance, remembering that multiple protocols can be sealed within ethernet envelopes. We may ask the router to open the ethernet envelopes and

© 2006 by Taylor & Francis Group, LLC

forward all NetWare (IPX) traffic to one network and all TCP/IP (IP) to another. In some cases, a certain protocol may require priority handling.

Multiprotocol Routers

Routers are made to read specific network layer protocols in order to maximize filtering and forwarding rates. If a router only has to route one type of network protocol, then it knows exactly where to look for destination addresses every time and can process packets much faster. However, realizing that different network layer protocols will have different packet structures with the destination addresses of various lengths and positions, some more sophisticated routers known as *multiprotocol routers* have the capability to interpret, process, and forward data packets of multiple protocols.

Some common network layer protocols and their associated network operating systems or upper layer protocols as well as other protocols which are actually data-link control protocols processed by some routers are listed in Table 22.1. Remembering that bridges are used to process data-link layer protocols, those routers that can also perform the functionality of a bridge are called bridging routers or *brouters*.

Router Configuration

Like bridges, routers generally take one of the two physical forms: (1) stand-alone variety, self-contained; and (2) modularized for installation in a slotted chassis. Routers may be installed to link LAN segments either locally or remotely. *Boundary routing* recognizes the need for simple, affordable wide area network devices at remote offices while providing full routing capabilities throughout the wide area network. Boundary routing's physical topology is sometimes referred to as a *hub and spoke topology* due to the fact that each remote branch is connected to a hub office via a single WAN link. If redundant links are a business requirement of a particular node, then it must be a full-function router and not a boundary router in this topology.

Full-function routers are placed at each hub or central node, while less sophisticated boundary routers, or branch-office routers, are placed at each remote or spoke node. Since there are only connected to a single WAN link, these boundary routers make only one decision when examining each piece of data, "If these data are not addressed to a local destination, then it should be forwarded." This forward-if-not-local logic should suggest that, in fact, these boundary routers are acting as bridges.

Gateways

Recalling that in terms of the OSI model, repeaters are considered a physical layer (layer 1) device, bridges are considered a data-link layer (layer 2) device, and routers are considered a network layer (layer 3) device, it could be said that *gateways* provide for interoperability on the session, presentation, and application layers (layers 5–7). Whereas repeaters, bridges and routers provide increasingly more sophisticated connection between two LANs, gateways provide transparent connection between two totally different computing environments. Specialized gateways also translate between different database management systems and are called *database gateways,* or between different e-mail systems and are called e-mail gateways.

The gateway is usually a computer with physical connections to both computing environments to be linked. In addition, the gateway also executes specially written software, which can translate messages between the two computing environments. Unlike other internetworking devices described, gateways are more concerned with translation than with processing destination addresses and delivering messages as efficiently as possible.

22.5 Wide Area Networks

Applications

Wide Area Network Architecture

To better understand all of the current and emerging wide area networking technologies and services, a simple model defining the major segments and interrelationships of an overall wide area network architecture is shown in Fig. 22.11. *User demands* are the driving force behind the current and emerging

© 2006 by Taylor & Francis Group, LLC

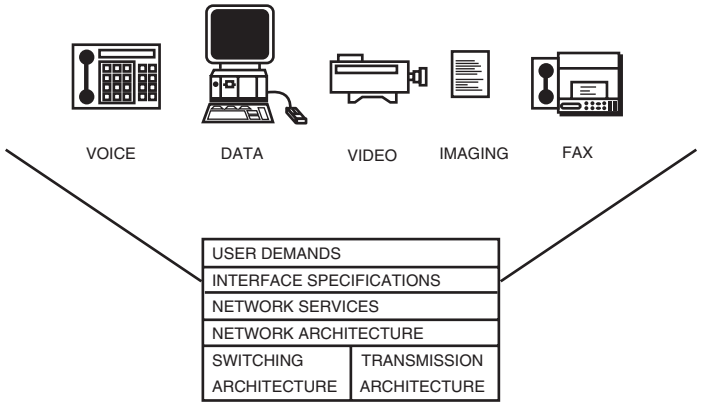


FIGURE 22.11 Major components of a wide area network architecture.

wide area *network services* which are offered to business and residential customers. Companies offering these services are in business to generate profits by implementing the underlying architectures that will enable them to offer the wide area networking services that users are demanding at the lowest possible cost.

Users are demanding simple, transparent access to variable amounts of bandwidth as required. In addition, this wide area network access must offer support for transmission of data, video, imaging, and fax as well as voice. One of the primary driving forces of increased capacity and sophistication for wide area network services is *LAN interconnection*.

Circuit Switching vs. Packet Switching

Switching of some type or another is necessary in a wide area network because the alternative is unthinkable. To explain, without some type of switching mechanism or architecture, every possible source of data in the world would have to be directly connected to every possible destination of data in the world, not a very likely prospect.

Circuit Switching

Switching allows temporary connections to be established, maintained, and terminated between message sources and message destinations, sometime called sinks in data communications. In the case of the voice-based phone network with which most people are familiar, a call is routed through a central office piece of equipment known as a switch, which creates a temporary circuit between the source phone and the phone of the party to whom one wishes to talk. This connection or circuit only lasts for the duration of the call. This switching technique is known as *circuit switching* and is one of two primary switching techniques employed to deliver messages from here to there. In a circuit switched network, a switched dedicated circuit is created to connect the two or more parties, eliminating the need for source and destination address information such as that provided by packetizing techniques.

The *switched dedicated circuit* established on circuit switched networks makes it appear to the user of the circuit as if a wire has been run directly between the phones of the calling parties. The physical resources required to create this temporary connection are dedicated to that particular circuit for the duration of the connection. If system usage should increase to the point where insufficient resources are available to create additional connections, users would not get a dial tone.

Packet Switching

The other primary switching technique employed to deliver messages from here to there is known as *packet switching*. Packet switching differs from circuit switching in several key areas. First, packets travel one at a time from the message source through a *packet switched network,* otherwise known as a *public data network,*

© 2006 by Taylor & Francis Group, LLC

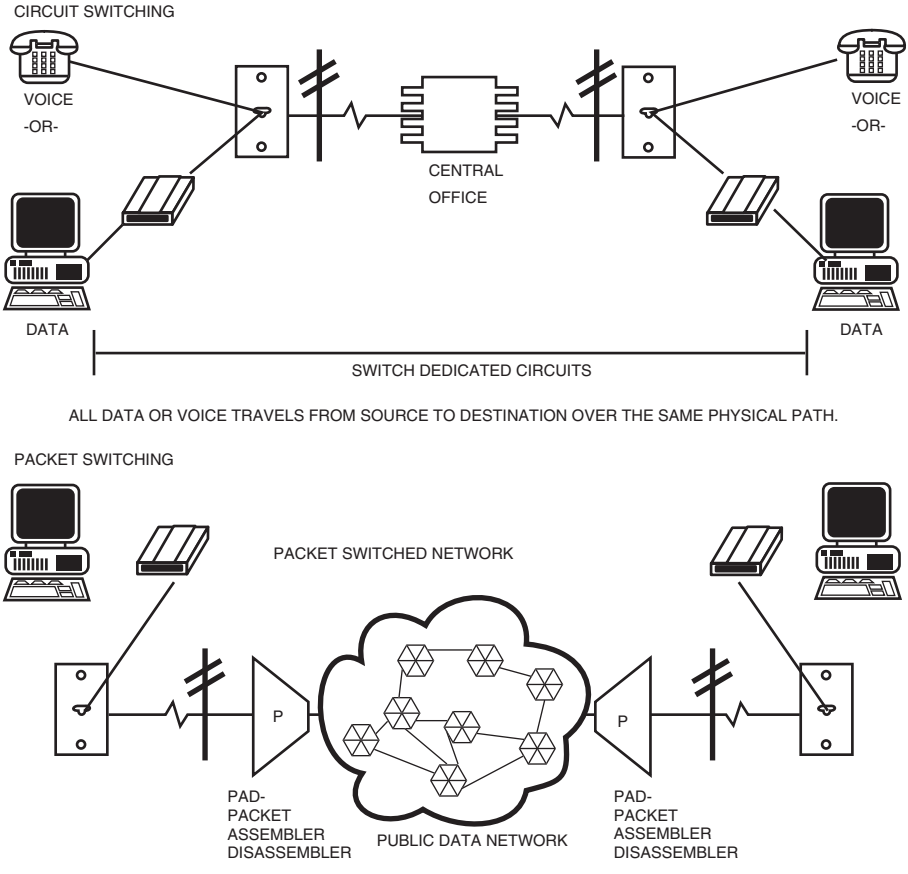


FIGURE 22.12 Circuit-switching vs. packet switching.

to the message destination. A packet switched network is represented in network diagrams by a symbol which resembles a cloud. Figure 22.12 illustrates such a symbol, as well as the difference between circuit switching and packet switching. The cloud is an appropriate symbol for a packet switched network since all that is known is that the packet of data goes in one side of the public data network (PDN) and comes out the other. The physical path which any packet takes may be different than other packets and in any case, is unknown to the end users. What is beneath the cloud in a packet switched network is a large number of *packet switches,* which pass packets among themselves as the packets are routed from source to destination.

Remember that packets are specially structured groups of data, which include control and address information in addition to the data itself. These packets must be assembled (control and address information added to data) somewhere before entry into the packet switched network and must be subsequently disassembled before delivery of the data to the message destination. This packet assembly and disassembly is done by a device known as a *packet assembler/disassembler (PAD)*. PADs may be stand-alone devices or may be integrated into modems or multiplexers.

These PADs may be located at an end-user location, or may be located at the entry point to the packet switched data network. Figure 22.12 illustrates the latter scenario in which the end users employ regular modems to dial up the value added network (VAN) or on-line information service, which provides the PADs to properly assemble the packets prior to transmission over the packet switched network.

© 2006 by Taylor & Francis Group, LLC

Packet Switched Networks

The packet switches illustrated inside the PDN cloud in Fig. 22.12 are generically known as *data switching exchanges (DSEs)*, or *packet switching exchanges (PSEs)*. DSE is the packet switching equivalent of the DCE (data communications equipment) and DTE (data terminal equipment) categorization for modems and dial-up transmission.

Another way in which packet switching differs from circuit switching is that as demand for transmission of data increases on a packet switched network, additional users are not denied access to the packet switched network. Overall performance of the network may suffer, errors and retransmission may occur, or packets of data may be lost, but all users experience the same degradation of service. This is because, in the case of a packet switched network, data travel through the network one packet at a time, traveling over any available path within the network rather than waiting for a switched dedicated path as in the case of the circuit switched network.

For any packet switch to process any packet of data bound for anywhere, it is essential that packet address information be included on each packet. Each packet switch then reads and processes each packet by making routing and forwarding decisions based on the packet's destination address and current network conditions. The full destination address uniquely identifying the ultimate destination of each packet is known as the *global address*.

Because an overall data message is broken up into numerous pieces by the packet assembler, these message pieces may actually arrive out of order at the message destination due to the speed and condition of the alternate paths within the packet switched network over which these message pieces (packets) traveled. The data message must be pieced back together in proper order by the destination PAD before final transmission to the destination address. These self-sufficient packets containing full source and destination address information plus a message segment are known as *datagrams*.

A Business Perspective on Circuit-Switching vs Packet Switching

If the top-down model were applied to an analysis of possible switching methodologies, circuit switching and packet switching could be properly placed on either the network or technology layers. In either case, in order to make the proper switching methodology decision, the top-down model layer directly above the network layer, namely, the data layer must be thoroughly examined. The key data layer question becomes: What is the nature of the data to be transmitted and which switching methodology best supports those data characteristics? The first data-related criteria to examine is the *data source*. What is the nature of the application program (application layer) which will produce this data? Is it a transaction oriented program or more of a batch update or file oriented program?

A transaction oriented program, producing what is sometimes called *interactive data*, is characterized by short *bursts* of data followed by variable length pauses due to users reading screen prompts or pauses between transactions. This *bursty transaction-oriented traffic*, best categorized by banking transactions at an automatic teller machine, must be delivered as quickly and reliably as the network can possibly perform. In addition to data burstiness, time pressures, and reliability constraints are other important data characteristics that will assist in switching methodology decision making.

Applications programs more oriented to large file transfers or batch updates have different data characteristics than transaction oriented programs. Overnight updates from regional offices to corporate headquarters or from local stores to regional offices are typical examples. Rather than occurring in bursts, the data in these types of applications are usually large and flowing steadily. These transfers are important, but often not urgent. If file transfers fail, error detection and correction protocols can retransmit bad data or even restart file transfers at the point of failure.

Defining Terms

Carrier sense multiple access with collision detection (CSMA/CD): A scheme for network communication flow.

Client: The end-users of a network and its resources, typically workstations or personal computers.

© 2006 by Taylor & Francis Group, LLC

Ethernet: A network architecture adhering to IEEE 802.3; a CSMA/CD-based architecture traditionally installed in a bus configuration, but more recently typically in a hub-based star physical topology.

Fiber distributed data interface (FDDI): A networking scheme using separate rings around which data move simultaneously in opposite directions to achieve high speed and operational redundancy.

Gateway: A network device designed to provide a transparent connection between two totally different computing environments.

Hub: The heart of a star physical topology, alternatively known as a concentrator, repeater, or multistation access unit (MAU).

Network interface card (NIC): The physical device or circuit used to interface the network with a local workstation or device.

Network management: The overall task of monitoring and analyzing network traffic and correcting network-related problems.

Open systems interconnection (OSI) model: A framework for organizing networking technology developed by the International Standards Organization.

Router: A device that reads specific network layer protocols in order to maximize filtering and forwarding rates on a network.

Server: The element of a network designed to facilitate and manage the sharing of resources among client devices and workstations.

Token ring: A network architecture adhering to IEEE 802.5, utilizing a star physical topology, sequential message delivery, and a token passing access methodology.

Wireless LAN: An emerging networking system utilizing radio or infrared media as the interconnection method between workstations.

References

Bachus, K. and Longsworth, E. 1993. Road nodes. *Corporate Computing* 2(3):54–61.
Bradner, S. and Greenfield, D. 1993. Routers: Building the highway. *PC Magazine* 12(6):221–270.
Derfler, F. 1993. Ethernet adapters: Fast and efficient. *PC Magazine* 12(3):191.
Derfler, F. 1993. Making the WAN connection: Linking LANs. *PC Magazine* 12(5):183–206.
Derfler, F. 1993. Network printing: Sharing the wealth. *PC Magazine* 12(2):249.
Derfler, F. 1993. To catch a thief. *PC Magazine* 12(16):NE1–NE9.
Derfler, F. 1994. Extend your reach. *PC Magazine* 13(14):315–351.
Derfler, F. 1994. Peer-to-peer LANs: Peer pressure. *PC Magazine* 13(8):237–274.
Donovan, W. 1993. A pain-free approach to SNA internetworking. *Data Communications* 22(16):99.
Gasparro, D. 1994. Putting wireless to work. *Data Communications* 23(5):57–58.
Goldman, J. 1995. *Applied Data Communications: A Business Oriented Approach*. Wiley, New York.
Greenfield, D. 1993. To protect and serve. *PC Magazine* 12(9):179.
Gunnerson, G. 1993. Network operating systems: Playing the odds. *PC Magazine* 12(18):285–333.
Harvey, D. and Santalesa, R. 1994. Wireless gets real. *Byte* 19(5):90.
Held, G. 1993. *Internetworking LANs and WANs*. Wiley, New York.
Held, G. 1994. *The Complete Modem Reference*. Wiley, New York.
Held, G. 1994. *Ethernet Networks: Design Implementation, Operation and Management*. Wiley, New York.
Held, G. 1994. *Local Area Network Performance Issues and Answers*. Wiley, New York.
Heywood, D. et al. 1992. *LAN Connectivity*. New Riders, Carmel, IN.
Jander, M. and Johnson, J. 1993. Managing high speed WANs: Just wait. *Data Communications* 22(7):83.
Johnson, J. 1993. LAN modems: The missing link for remote connectivity. *Data Communications* 22(4):101.
Johnson, J. 1994. Wireless data: Welcome to the enterprise. *Data Communications* 23(5):42–55.
Karney, J. 1993. Network lasers: Built for speed. *PC Magazine* 12(20):199.
Madron, T. 1993. *Peer-to-Peer LANs: Networking Two to Ten PCs*. Wiley, New York.
Mandeville, R. 1994. Ethernet switches evaluated. *Data Communications* 23(4):66–78.
Mathias, C. 1994. New LAN gear naps unseen desktop chains. *Data Communications* 23(5):75–80.

© 2006 by Taylor & Francis Group, LLC

Peterson, M. 1993. Network backup evolves. *PC Magazine* 12(16):277–311.
Pompili, T. 1994. The high speed relay. *PC Magazine* 13(1):NE1–NE12.
Quiat, B. 1994. V.FAST, ISDN, or switched 56 K. *Network Computing* 5(3):70.
Raskin, R. 1993. Antivirus software: Keeping up your guard. *PC Magazine* 12(5):209–264.
Rosen, B. and Fromme, B. 1993. Toppling the SNA internetworking language barrier. *Data Communications* 22(9):79.
Saunders, S. 1993. Choosing high speed LANs: Too many technologies, too little time? *Data Communications* 22(13):58–70.
Saunders, S. 1994. Building a better token ring network. *Data Communications* 23(7):75.
Saunders, S. 1994. Full duplex ethernet: More niche than necessity? *Data Communications* 23(4):87–92.
Schlar, S. 1990. *Inside X.25: A Manager's Guide*. McGraw–Hill, New York.
Shimada, K. 1994. Fast talk about fast ethernet. *Data Communications* 23(5):21–22.
Stallings, W. 1992. *ISDN and Broadband ISDN,* 2nd ed. Macmillan, New York.
Stevenson, T. 1993. Best of a new breed: Groupware—Are we ready? *PC Magazine* 12(11):267–297.
Tabibian, O.R. 1994. Remote access: It all comes down to management. *PC Magazine* 13(14):NE1–NE22.
Thomas, R. 1994. PPP starts to deliver on interoperability promises. *Data Communications* 23(6):83.
Tolly, K. 1993. Can routers be trusted with critical data? *Data Communications* 22(7):58.
Tolly, K. 1993. Checking out channel-attached gateways. *Data Communications* 22(8):75.
Tolly, K. 1993. Token ring adapters: Evaluated for the enterprise. *Data Communications* 22(3):73.
Tolly, K. 1994. FDDI adapters: A sure cure for the bandwidth blues. *Data Communications* 23(10):60.
Tolly, K. 1994. How accurate is your LAN analyzer? *Data Communications* 23(2):42.
Tolly, K. 1994. The new branch-office routers. *Data Communications* 23(11):58.
Tolly, K. 1994. Testing dial up routers: Close, but no cigar. *Data Communications* 23(9):69.
Tolly, K. 1994. Testing remote ethernet bridges. *Data Communications* 23(8):81.
Tolly, K. 1994. Testing remote token ring bridges. *Data Communications* 23(6):93.
Tolly, K. 1994. Testing UNIX-to-SNA gateways. *Data Communications* 23(7):93.
Tolly, K. 1994. Wireless internetworking. *Data Communications* 22(17):60.

Further Information

Books:

The following two books are excellent additions to a professional library for overall coverage of data communications and networking.

Newton, H., *Newton's Telecom Dictionary,* Telecom Library, New York.
Goldman, J. E., *Applied Data Communications: A Business Oriented Approach,* Wiley, New York.

© 2006 by Taylor & Francis Group, LLC

23

Printing Technologies and Systems

23.1 Introduction **23**-1
Resolution and Addressability • Grayscale • Dot Microstructure
23.2 Printing Technologies **23**-3
23.3 Nonimpact Printing Technologies **23**-4
Ink Jet • Continuous Ink Jet • Drop On Demand (DOD) Ink Jet • Thermal Ink Jet/Bubble Jet DOD Printers • Piezoelectric DOD Printers • Grayscale Methods for DOD Ink Jet Printers • Ink and Paper for Ink Jet Devices
23.4 Thermal Printing Technologies **23**-11
Direct Thermal • Direct Thermal Transfer • Dye Diffusion Thermal Transfer • Resistive Ribbon Thermal Transfer
23.5 Electrophotographic Printing Technology **23**-13
Printing Process Steps • Dot Microstructure
23.6 Magnetographic and Ionographic Technologies **23**-15
23.7 System Issues **23**-16
Color Management

John D. Meyer

23.1 Introduction

The basic parameters of print quality are **resolution**, **addressability**, **gray scale**, and dot microstructure. A real device also has intrinsic variability in the printing process, producing visual artifacts, which come under the general heading of *noise*. Some of the more common manifestations of this are background scatter, dot placement errors, voids (due to nozzle malfunction in ink jet, for example), and banding in images. The significance of any of these aspects of print quality can only be determined by examining them with respect to the properties of the human visual system. The design choices of the basic print quality parameters are, therefore, guided by the properties of the human visual system to determine where improvement needs to be made or where little is to be gained by increasing any one of the specifications.

Resolution and Addressability

Resolution, the most widely used specification to rate print quality, is sometimes confused with the related term addressability. Fundamentally, resolution refers to the ability of the device to render fine detail. This simple definition is complicated by the fact that detail can be regarded as the fineness of the width of a line, the transition between white paper and printed intensity, and/or the smoothness of the edge of a curved line or a line printed at any arbitrary angle. In the simplest case, the resolution of a printer is defined as

© 2006 by Taylor & Francis Group, LLC

the spacing of the dots such that full coverage is obtained, that is, no white paper can be seen. For circular dots placed on a square grid, this number would be calculated by dividing the diameter by the square root of two and taking its inverse. For example, an ideal 300 dots per inch (dpi) printer would produce 120-μm-diam dots at an 85 μm spacing. In practice, the dot would be made somewhat larger to allow for dot placement errors. This definition is best understood in terms of the finest line that can be printed by the device. At 300 dpi the line would exhibit a perceptible edge waviness, especially when printed at certain sensitive angles. This would also be true of curved lines. In addition, the range of lines of increasing thickness would have discontinuities since they would consist of an integral number of the basic line, each spaced at 85 μm. These issues have an important bearing on text print quality, which depends on the ability to render both curved and straight lines at variable widths.

The preceding definition is related to the specification of resolution with respect to the human visual system. In this case resolution is determined by the closeness of spacing between alternate black and white lines of equal width and defined contrast. These are known as *line pairs* and for a 300-dpi printer it would result in a value of 150 line pairs per inch. This is not strictly correct since the black lines would be wider than the white spaces due to the roundness of the dot. Since the human visual system has a response that approaches zero near 300 line pairs per inch, gains made in text print quality by increasing resolution alone can be expected to diminish above this value. At this point, issues such as print noise and grayscale enter if further improvement in print quality is desired.

To focus only on the resolution as defined in the previous paragraphs ignores the specific needs of the components of the printed material, that is, text and lines vs. images and area fill. Gains in text print quality may be had if the device can space the dots closer than the fundamental resolution. This can result in substantial dot overlap but allows the line width to be varied more continuously. In addition, at the edge of a curved line, the subpixel adjustments of individual dots increase the perception of smoothness commonly known as getting rid of the *jaggies*. This ultimate dot spacing of the device is called addressability. For example, printers employing this technique are specified as 300 $\times$ 600 dpi indicating a native resolution of 300 dpi in the horizontal direction and a vertical addressability of 600 dpi.

Grayscale

The ability of a printing technology to modulate the printed intensity on the page is referred to as its grayscale capability. There are three ways in which this may be accomplished: variation of the dot size, variation of the intensity of the printed dot, and digital **halftoning** techniques. The first two depend on the intrinsic properties of the technology, whereas digital halftoning can be employed by any printer. A printer that can continuously vary its intensity from white paper through to maximum colorant density is described as having *continuous tone capability*. Other technologies produce a modest number of intensity levels and make use of digital halftoning techniques to create a continuous tone effect. The manner in which gray scale is achieved is of obvious importance in image printing, particularly in the case of color. In recent years considerable effort has gone into the development of sophisticated digital halftoning algorithms to enable binary (single dot size and no intensity modulation) printers to render images. The resulting image quality depends more strongly on resolution than addressability. But the impact of even a few intrinsic gray levels on the print quality achieved by these algorithms can be dramatic.

An important parameter in grayscale considerations is that of the dynamic range, which is simply called range in the graphic arts. This is measured in terms of optical density, the negative logarithm of the reflectance. An optical density of 1.0 represents 10% of reflected light per instant flux, an optical density of 2.0 corresponds to 1% reflectance, and so on. For printed material the smoothness of the printed surface limits the maximum optical density obtainable. If the surface is smooth and mirrorlike, then the print appears glossy and can have optical densities approaching 2.4. The smooth surface reflects light in a specular manner and, therefore, scatters little stray light from the surface into the eye, and the color intensity is not desaturated. It is most noticeable in the case of photographic paper that has a high gloss finish. If the optical density range of the print is high, it is said to have high dynamic range and a very

© 2006 by Taylor & Francis Group, LLC

pleasing image will result. Not all papers, however, are designed to have a glossy finish. Papers used in the office are also used in copiers and have a surface which produces diffuse reflection at the interface between the air and the paper. For most uncoated, nonglossy papers this will be between 3–4% and limits the maximum optical density to around 1.4. Image quality on these stocks will depend on the fixing of the colorant to the substrate to produce a smooth surface. The potential image quality for a printer is therefore a complex tradeoff involving the design choices of resolution, addressability, grayscale method, **digital halftoning** algorithm, paper stock, colorant, and fixing technology. Conclusion: for images, resolution alone is not a predictor of print quality.

Dot Microstructure

The microscopic nature of the dot produced by a given technology also has a bearing on final print quality. The most important parameter here relates to the edge gradient of the dot. Known as the *normal-edge profile,* it characterizes the transition between white paper and maximum colorant intensity, that is, the gradient of optical density that occurs at the edge of the dot and measures the steepness of the transition from white paper to full optical density. Some technologies, such as electrophotography, can vary this profile by adjusting various parameters in the imaging and developing process. For ink jet, various paper types will produce different normal-edge profiles. If the profile is very steep, that is, the transition occurs in a very small distance such as 5 μm, then the dot is described as being a hard dot or having a very sharp edge. This is desirable when printing lines and text that benefit from very sharp transitions between black and white. If this transition is gradual, the dot is described as being soft and produces a blurring of the edge, which can degrade the text quality. In the case of images, where smooth tones and tonal changes are desired, a soft dot can be very beneficial.

Hybrid Methods

From what has been said it should not be inferred that the needs of texts and images are in opposition. In recent years the intrinsic grayscale capability has been used to advantage in improving text print quality. The removal of jaggies can be greatly assisted by the combination of increased addressability and a few gray levels. By the use of gray levels in the region of the jagged stairstep, the transition can be made to take place over several pixels. This is, in essence, blurring the transition to make it less visible to the eye. In the case of certain fonts, there is fine detail requiring resolutions greater than the native resolution of the printer. This fine detail can be rendered through a combination of gray levels and regular pixels. The implementation of these methods requires a complex set of rules to be applied to the data bit stream before it is sent to the marking level of the printer. These rules draw heavily on image processing techniques and a knowledge of the human visual system and are proprietary. Skillfully applied they can have a dramatic effect on the text and line quality. There are a variety of trademarked names for these technologies designed to convey the sense of enhancement of the print quality.

The application of image processing techniques to manipulate the intrinsic properties of electronic printing technologies have made resolution an insufficient measure of print quality. A more comprehensive measure is needed to simplify the identification of the printing technology to serve the design goals for final output quality. Until such a metric is devised, the tradeoff analysis just described, implemented by means of industry standard test charts that separately probe the printer properties, will provide a predictive measure of print quality. Such test charts must also contain test images, which will be subject to the proprietary subjective image enhancement algorithms offered by the manufacturer.

23.2 Printing Technologies

The four basic elements of any printing technology are: addressing, marking substance and its storage and delivery, transfer of the marking substance, and fixing. Addressing refers to the communication of electronic data to the marking unit, typically via electronic or optical means. The marking substance contains the

© 2006 by Taylor & Francis Group, LLC

colorant, vehicle/carrier material for transport, binders to secure the colorants to the paper, stabilizing agents to resist fading, and technology specific additives such as biocides for liquid inks. The transfer process is the fundamental physical mechanism whereby a specific amount of the marking substance is removed from the bulk and transferred to the paper. Fixing embodies the processes of adhesion, drying, or solidification of the material onto the paper to form a durable image. These fundamental subsystems interact with each other to give each printing technology its own unique characteristics. The common classification of printing technologies today begins with the broad separation into two classes: Impact and nonimpact printing technologies. Impact methods achieve transfer via the direct mechanical application of force or pressure via a marking element, which can be either a fine wire or fully formed character onto a colorant carrying ribbon in contact with the paper; the simplest form of this is a typewriter. Nonimpact methods cover a wide range of technologies that achieve transfer through a variety of means that may be either contact or noncontact in nature.

23.3 Nonimpact Printing Technologies

Ink Jet

The transfer process of ink jet printing is one of removing a drop of liquid ink from the bulk and giving it a velocity of sufficient precision and magnitude to place it on a substrate in close proximity to but not touching the printhead. There are three broad techniques: *continuous, electrostatic,* and *drop on demand*. Continuous ink jet printing, because of its intrinsic high drop rate, has tended to find more applications in commercial systems; electrostatic methods have yet to find widespread application, but have been used for facsimile recording; drop on demand, because of its simplicity and ease of implementation of color, has been widely accepted in the office and home market.

Continuous Ink Jet

The basic principle of continuous ink jet is to take advantage of the natural breakup process due to an instability in the jet that is formed when fluid is forced under pressure through a small orifice. This results from the interplay of surface tension and viscosity and takes place in a quasirandom manner unless external stimulation is applied. This breakup process was first studied by Rayleigh who characterized it via a growth rate for the instability, which depended on the jet diameter D, its velocity V, and the frequency F of any external stimulation. Rayleigh showed that the frequency for maximum growth rate of the instability was

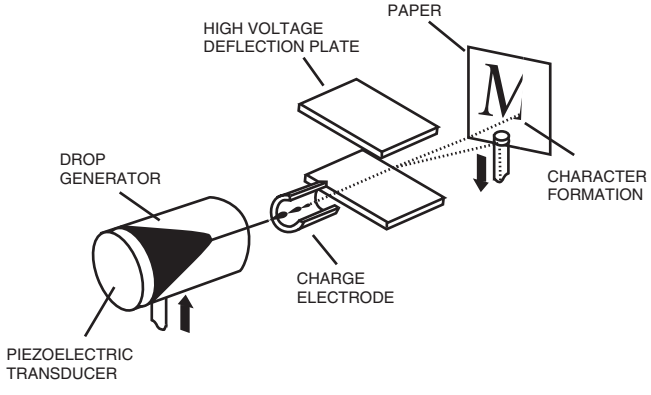


FIGURE 23.1 Character printing with continuous ink jet. The deflection plate applies an analog voltage to steer the drop to the desired location; unwanted droplets are undeflected and captured by the return gutter. (*Source:* Durbeck, R.C. and Sherr, S. 1988. *Hardcopy Output Devices.* Academic Press, San Diego, CA. With permission.)

© 2006 by Taylor & Francis Group, LLC

$F = V/4.5D$. By stimulating the jet at this frequency it is possible to obtain a uniform stream of droplets. The typical method of providing this stimulation today is via a piezoelectric transducer as an integral part of the printhead.

To make use of these droplets for printing it is necessary to charge them at *breakoff*. This is accomplished by placing electrodes in proximity to the breakup region of the jet. Deflection voltages are then applied farther downstream to direct the droplet to the substrate or into a collector for recirculation and reuse. The earliest techniques involved charging the droplet and applying a variable deflection field to direct it to a specific spot on the paper, enabling full height characters to be printed in one pass (see Fig. 23.1). Later methods focused on producing a stream of charged droplets and using the printing (high-voltage) electrode to deflect unwanted drops for recirculation and reuse (Fig. 23.2). This technique, known as *binary charged continuous ink jet,* lends itself to the construction of multiple nozzle arrays, and there are a number of page-wide implementations in use.

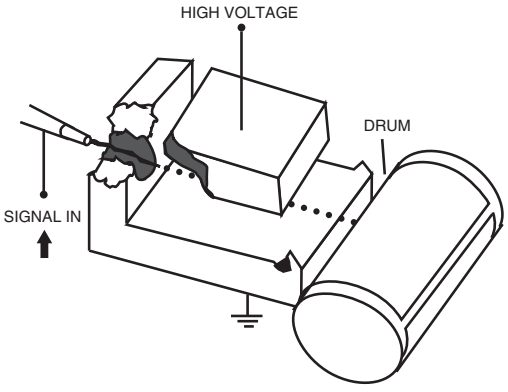


FIGURE 23.2 Binary charged continuous ink jet. Droplets, charged at breakoff, are directed to the paper on a rotating drum. Droplets not selected for printing are diverted by the high-voltage electrode to be collected by a return gutter. (*Source:* Durbeck, R.C. and Sherr, S. 1988. *Hardcopy Output Devices.* Academic Press, San Diego, CA. With permission.

Binary charged continuous ink jet with its high droplet rate makes a simple gray scaling technique possible. The dot on the paper is modulated in size by printing from one up to N droplets at the same location, where N is the number of different dot sizes desired. By operating at high frequencies and small drop volumes it is possible to produce sufficient gray levels such that full grayscale printing is achieved at reasonable print speeds. The most recent implementation of this method offers 512 gray levels at addressabilities between 200–300 pixels/in. To achieve the small fundamental droplet size, typical implementations employ glass capillaries with diameters of the order of 10 μm and are pressurized from 500–700 lb/in^2. A color printhead will contain four capillaries, one for each color ink plus black.

Drop On Demand (DOD) Ink Jet

For office and home applications the complexities of continuous ink jet technology, such as startup and shutdown procedures, ink recirculation, and the limited nozzle count, have led to the development of drop on demand ink jet technology. These devices employ unpressurized ink delivery systems and, as implied by their name, supply a drop only when requested. The basic technique employed is to produce a volume change in either the ink supply channel or an ink chamber adjacent to the nozzle such that the resulting pressure wave causes drop ejection. Refill is achieved by capillary forces and most DOD systems operate with a slight negative pressure at the ink reservoir. The mechanism for generating the pressure wave dominates the design of these devices, and there are two techniques extant in common DOD printers. One employs the pressure pulse derived from the vaporization of superheated fluid, and the other makes use of piezoelectric materials, which can be deformed by the application of electric potentials.

Devices employing the vaporization of superheated fluid are known concurrently as thermal ink jet or bubble jet printers, the choice of name depending on the manufacturer. Since drop on demand ink jets rely on capillary refill, their operational frequencies are much lower than for continuous ink jet devices. This stresses the importance of the compactness of the actuating system so as to achieve reasonable printing speeds via multiple nozzle printheads. The nozzles must also be precisely registered with respect to each other if systematic print artifacts are to be avoided.

© 2006 by Taylor & Francis Group, LLC

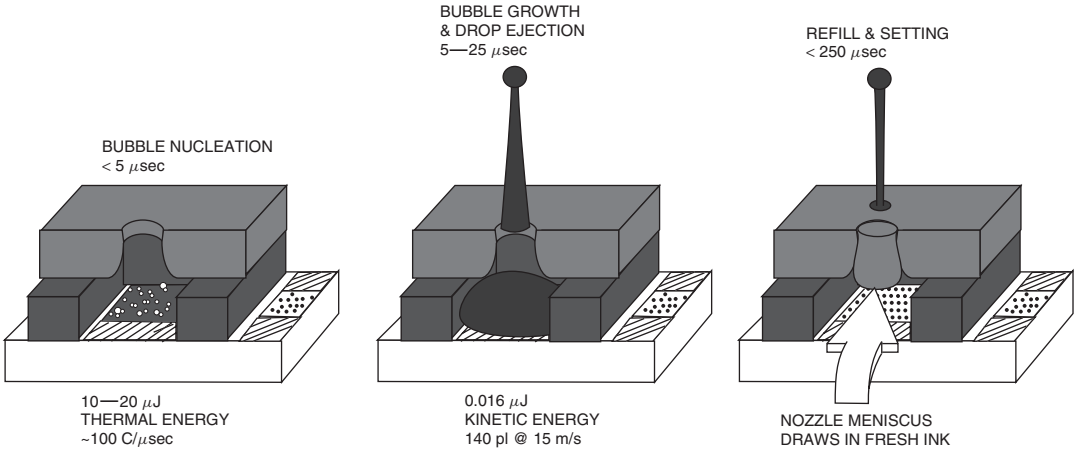


FIGURE 23.3 Drop ejection sequence for thermal ink jet. Nominal time frames and values for various parameters are given to indicate the scale of the three processes of nucleation, bubble growth, and jet formation followed by drop ejection and refill.

Thermal Ink Jet/Bubble Jet DOD Printers

When fluids are heated at extreme rates (e.g., 500×10^6 W/m^2), they enter a short-lived metastable state where temperatures can far exceed the boiling point at atmospheric pressure. The difference between the elevated temperature and the boiling point is known as the *degree of superheat*. This process does not continue indefinitely, and all fluids have what is known as a *superheat limit*. At this point nucleation and vaporization will occur in the bulk of the fluid. These devices employ an electrically driven planar heater (typically, 50–60 μm^2) in contact with the fluid. Under these conditions vaporization commences at the surface of the heater due to the presence of nucleation sites such as microscopic roughness. With correctly chosen heating rates this can be made very reliable. These heating rates lead to electrical pulse widths of 3–5 μs. In this time frame only a submicron portion of the fluid will be superheated. The net result is a vaporization pulse well in excess of atmospheric pressure and of approximately 3/4-μs duration. By locating a nozzle directly over or alongside the resistor this pressure pulse will eject a droplet (Fig. 23.3).

Within limits of the drop volume desired, it is found that the linear dimensions of the nozzle diameter and planar resistor are comparable. The actuator is therefore optimally compact, and this enables high-nozzle count printheads. The fabrication of the resistors is accomplished by photolithographic techniques common to the IC industry and the resistor substrates are silicon with a thin layer of insulating silicon dioxide. Precise registration from nozzle to nozzle is guaranteed under these circumstances, and electrical drive circuits may be integrated into the head to provide multiplexing capability. This is a valuable attribute for scanning printheads, which employ a flexible printed circuit for interconnect. These features have produced printheads currently numbering 300 or more nozzles for a single color. An additional benefit of the compactness of this technology is that the ink supply can be fully integrated with the printhead. This provides the user with virtually maintenance free operation as the printhead is replaced when the ink supply is consumed. Since the majority of problems arise from paper dust particles finding their way into a nozzle and occasionally becoming lodged there, printhead replacement provides for user service at reasonable cost. Some implementations feature a semipermanent printhead, which is supplied by ink from replaceable cartridges. The design choice is then a tradeoff involving many factors: frequency of maintenance, cost of operation, how often the printer is to be used, type of material printed, etc. The important subject of ink and paper for these printers will be taken up at the end of the section on DOD technologies.

© 2006 by Taylor & Francis Group, LLC

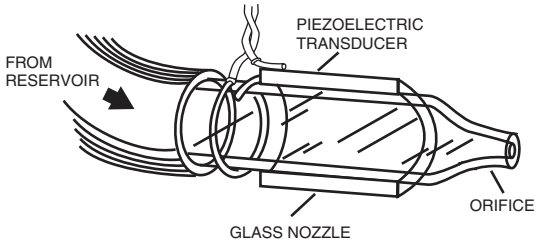


FIGURE 23.4 Squeeze tube piezoelectric ink jet. Early implementation of piezoelectric transducers for drop ejection. (*Source:* Advanced Technology Resources Corporation.)

Piezoelectric DOD Printers

Crystalline structures, which develop a spontaneous dipole moment when mechanically strained, thereby distorting their crystal structures, are called *piezoelectric*. These materials may conversely be caused to be distorted via electrical potentials applied to the appropriate planes of the cyrstal. Piezoceramics have a polarization direction established during the manufacturing process, and the applied fields then interact with this internal polarization to produce mechanical displacement. Depending on the direction of the applied fields, the material can compress or extend longitudinally or transversely. These materials have found widespread use as transducers for DOD printers. An early form was that of a sleeve over a glass capillary, which terminated in a nozzle (Fig. 23.4). Depending on the location of the electrodes either a radial or longitudinal compression could be applied leading to a pressure wave in the enclosed ink sufficient to eject a droplet. Using the diameter of the nozzle as a unit of linear dimension, this approach placed the transducer well upstream from the nozzle (Fig. 23.5). Implementation of this design in a multinozzle printhead required careful matching of transducers and fluid impedance of the individual channels feeding each nozzle. This was a challenging task, and most designs bond a planar transducer to an ink chamber adjacent to a nozzle, as shown in Fig. 23.6.

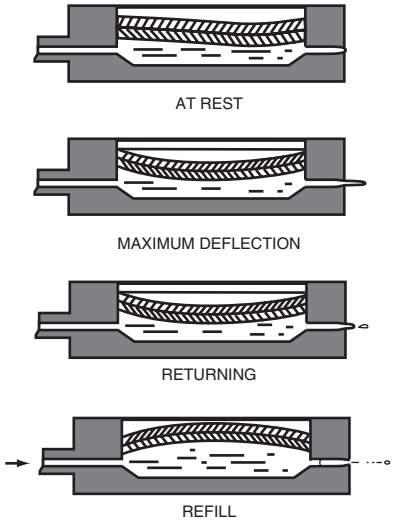


FIGURE 23.5 Drop ejection sequence for piezoelectric printhead. Schematic of drop ejection via deflection of piezocrystal bonded to an ink capillary. In practice the piezodrivers were located well upstream of the nozzle due their size. (*Source:* Advanced Technology Resources Corporation.)

© 2006 by Taylor & Francis Group, LLC

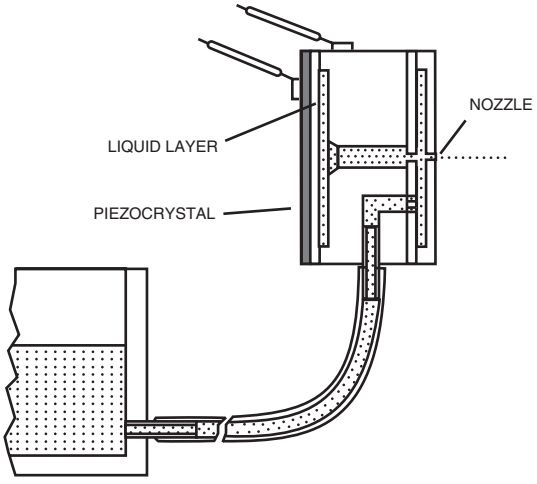


FIGURE 23.6 Design of Stemme Larson electric-driven DOD ink jet. Note the direct coupling of the pressure pulse to the ink chamber at the nozzle. (*Source:* Advanced Technology Resources Corporation.)

The method of directly coupling the piezoelectric transducer through an ink chamber to an exit nozzle has seen many enhancements and developments since its invention. A feature of some designs is that of air flow channeled at the orifice in such a way as to entrain the droplet as it exits the nozzle and to improve its directional stability, as well as to accelerate the droplet. This enables the device to be operated at lower transducer deflections and, therefore, at higher droplet rate since the settling time of the device has been reduced. Piezodevices can operate at elevated temperatures and are used to eject inks that are solid at room temperature. For solid inks the material is melted to a design temperature for appropriate viscosity and surface tension and then supplied to the piezoelectric-driven ink chamber. The ink then solidifies instantly on contact with the substrate.

A more recent innovation employs piezoelectric transducers operated in the longitudinal mode. The transducers are formed from a single block of piezoceramic material in the form of an array of rods. Suitably placed potentials excite the rods to extend in the longitudinal direction. By bonding one end of the rod in contact with a thin membrane forming the base of an ink chamber, a pressure pulse is generated similar to that of the previous design (Fig. 23.7). To achieve sufficient pressure amplitude a diaphragm is used that is substantially larger than the orifice exit diameter. The consequence of this is that high nozzle

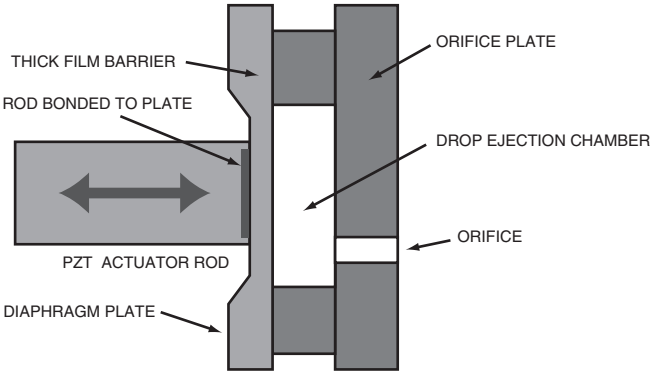


FIGURE 23.7 Schematic of ink chamber and actuator.

© 2006 by Taylor & Francis Group, LLC

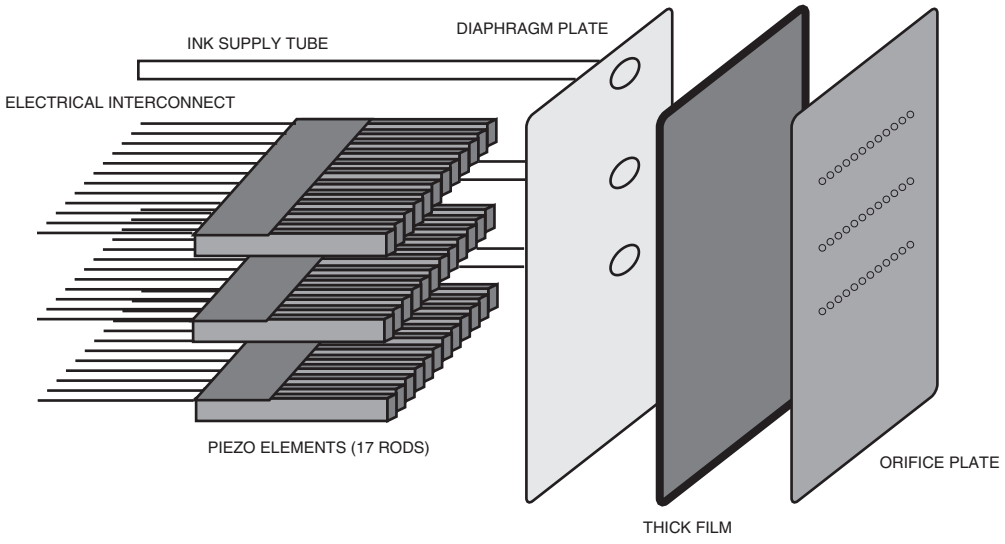


FIGURE 23.8 Exploded view of a multi chamber pen.

density printheads will require multiple rows of nozzles (Fig. 23.8). This design has been implemented to date with liquid inks only.

Grayscale Methods for DOD Ink Jet Printers

The drop rates for DOD devices are typically an order of magnitude less than those of continuous, pressurized systems. This dictates different strategies for the achievement of grayscale. Techniques are based on the generation of a few gray levels that, when incorporated into digital halftoning algorithms, such as error diffusion, clustered, dispersed dot, or blue-noise dither, produce a satisfactory grayscale. The number of levels necessary, their position relative to the maximum modulation achievable (i.e., maximum dot size or maximum intensity), and the specialized techniques employed in digital halftoning are an area of active research. There are many patents in the literature governing these techniques, and manufacturers seek to distinguish their devices by the technique offered. When combined with resolution enhancement methods mentioned in the section on print quality, printers with medium resolution, such as 300 dpi and 2 bits of grayscale, can produce remarkable results for both images, text, and graphics.

There are several methods available for DOD devices to modulate either the size or intensity of the dot. For piezodevices, pulse width modulation has been shown to produce droplets of different volumes and, therefore, dot sizes. All DOD ink jet devices have the option of ejecting a droplet repeatedly at the same location by passing over the same swath as many times as desired but this will affect throughput rates. Printheads with sufficient nozzle count can do this and still keep throughput rates within reason. For vapor bubble driven devices, a unique option exits by virtue of the short duration of the actuating bubble. Typical lifetime of bubbles in these devices, from vaporization through to bubble collapse, is of the order of 20 μs. If the resistor is pulsed shortly after bubble collapse, a second droplet can be ejected virtually on the tail of the initial droplet. This technique has been called *multidrop* in the literature. The ink chamber is fired under partial refill conditions, but with proper design several droplets can be ejected by this method at drop rates at around 40 kHz and having substantially the same volume (Fig. 23.9). These merge on the substrate to produce different dot sizes according to the number of droplets ejected for the location. This is not an option for most piezodevices due to the slower settling time of the actuator. Dye dilution methods have also been demonstrated as a way of modulating the intensity of the dot. If no halftone algorithm is employed, this will require many sets of nozzles to accommodate the different dye dilutions.

© 2006 by Taylor & Francis Group, LLC

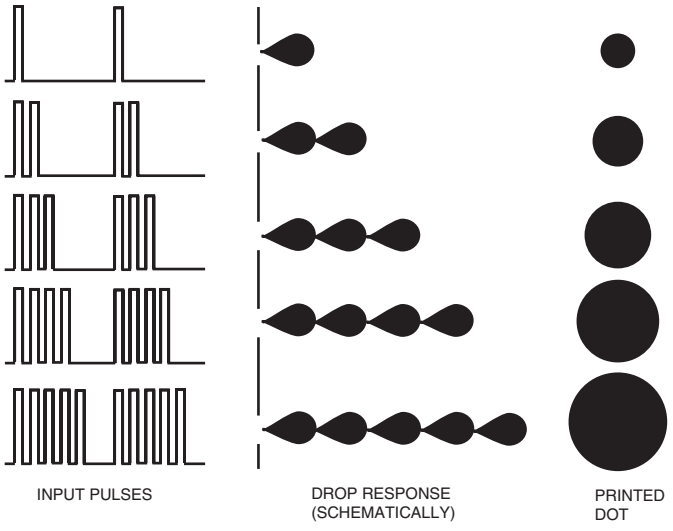


FIGURE 23.9 Stylized representation of multidrop process. Each input pulse consists of a group of drive pulses chosen for the final dot size desired. (*Source:* Durbeck, R.C. and Sherr, S. 1988. *Hardcopy Output Devices.* Academic Press, San Diego, CA. With permission.)

Ink and Paper for Ink Jet Devices

When liquid inks are employed the paper properties have a major impact on the print quality. The ink droplets will be absorbed by a substrate whose internal structure and surface energy will determine the size, shape, and overall microstructure of the drop. Paper, being a interlocking mesh of cellulose fibers with sizing and binding chemistry, is quite variable in nature. Figure 23.10 is a schematic indication of the response of paper to different volumes of ink. Note that it can be very nonlinear at low drop volumes and either flat or high gain at large volumes. The implication is that by simply changing the paper the print quality is altered. To control this variablity some papers are given a thin coat of claylike material containing whiteners, which are often fluorescent. This coating presents a microporous structure that is more uniform than the cellulose fibers.

Dot formation on coated papers is therefore circular and more stable than on uncoated stock. Uncoated papers allow the ink to wick down the fibers producing an effect known as *feathering* of the dot. In this case, microscopic tendrils of dye appear at the edge of the dot giving it and the overall print quality a blurred effect. This is particularly serious in the case of text printing, which benefits most from sharp dot edges. Feathering is common for papers used in xerographic copiers. Bond paper, which is a popular office stock, is a partially coated paper and exhibits little feathering.

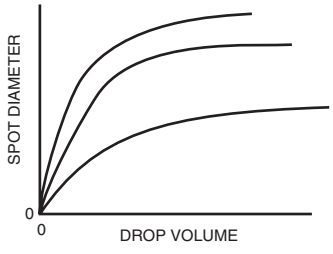


FIGURE 23.10 Paper response curves. The lower curve is typical of coated stock, which minimizes dot spread. (*Source:* Durbeck, R.C. and Sherr, S. 1988. *Hardcopy Output Devices.* Academic Press, San Diego, CA. With permission.)

Depending on the manufacturer, several techniques are employed to minimize the impact of paper variability on print quality. One method, the use of a heater, takes advantage of the fact that absorption into the paper does not commence immediately upon contact. There is a period known as the *wetting time*, which can be as long as 80 μs during which no absorption takes place. The application of heat immediately in the vicinity of the printhead swath can effectively "freeze the dot" by vaporizing carrier. This makes

© 2006 by Taylor & Francis Group, LLC

the printer insensitive to change in paper stock and provides uniformly high print quality regardless of substrate. Other methods make use of altering the fluid chemistry by means of surfactants, which increase penetration rate, and high vapor pressure additives to increase removal of fluid into the atmosphere. If the ink penetrates quickly then it is less likely to spread sideways and, thereby, dot size variation is lessened. When choosing a drop on demand ink jet printer, it is advisable to test the performance over the range of paper stocks to be used. In some cases it will be found that high-quality printing can only be obtained when a paper specified by the manufacturer is chosen.

With reference to the section on print quality, it should be kept in mind that the choice of paper will affect the overall dynamic range of print. Text printing, to be pleasing, needs to have an optical density of at least 1.3–1.4. For images, the more dynamic range the better, and special coated stock will always excel over copy paper if image quality is important. Many of the coated papers available still have a matte surface that diffusely reflects the light and limits the dynamic range for reasons previously discussed. Some manufacturers now offer a high-gloss substrate specifically intended for images. These substrates have a plastic base with special coatings to absorb the ink through to the substrate leaving a high gloss finish. This greatly improves the dynamic range to the point of approximating that of photographic paper. These substrates provide ink jet printers with the capability to produce highly saturated brilliant colors with exceptional chromatic and dynamic range and should be used if image printing is the primary objective.

Besides excellent print quality there are other demands placed on the ink. It must provide reliable operation of the device and a durable image. By this it is meant that the image does not fade rapidly, that it is mechanically sound, that it cannot be easily removed from the paper, and that it is impervious to liquids. For liquid ink, this is a challenge since solvent-based color highlighter pens are commonly used to mark up printed documents. These solvents can cause the ink to smear depending on the choice of ink chemistry and the manner in which the colorants are fixed to the substrate. These issues focus on colorant chemistry, and much research is applied to this problem. There are fade-proof dyes, but many are either incompatible with the ink vehicle, typically water, or are toxic or mutagenic.

23.4 Thermal Printing Technologies

Printing technologies that employ the controlled application of thermal energy via a contacting printhead to activate either physical or chemical image formation processes come under this general classification. There are four thermal technologies in current use: direct thermal, direct thermal transfer, dye diffusion thermal transfer, and resistive ribbon thermal transfer.

Direct Thermal

This is the oldest and most prolifically applied thermal technology. The imaging process relies on the application of heat to a thermochromic layer of approximately 10 μm in thickness coated onto a paper substrate. The thermally active layer contains a leuco dye dispersed along with an acid substance in a binder. Upon heating fusion melting occurs resulting in a chemical reaction and conversion of the leuco dye into a visible deeply colored mark. Key to this process is the design of the printhead, which can be either a page-wide array or a vertical array or a scanning printhead. Two technologies are in use, thick film and thin film. Thick-film printheads have resistor material between 10 and 70 μm. The resistive material is covered with a glass layer approximately 10 μm thick for wear resistance. The thin-film printheads bear a strong resemblance to those found in thermal ink jet printheads. They employ resistive material, such as tantalum nitride, at $\frac{1}{10}$ μm thickness and a 7-μm-thick silicon dioxide wear layer. Thin-film heads are manufactured in resolutions up to 400 dpi. In each case the resistors are cycled via electrical heating pulses through temperature ranges from ambient (25°C) up to 400°C. Overall, the thin-film printheads excel in energy efficiency conversion, print quality, response time, and resolution. For these reasons the thin-film printheads are used when high resolution is required, whereas the thick-film printhead excels in commercial applications such as bar coding, airline tickets, fax, etc.

© 2006 by Taylor & Francis Group, LLC

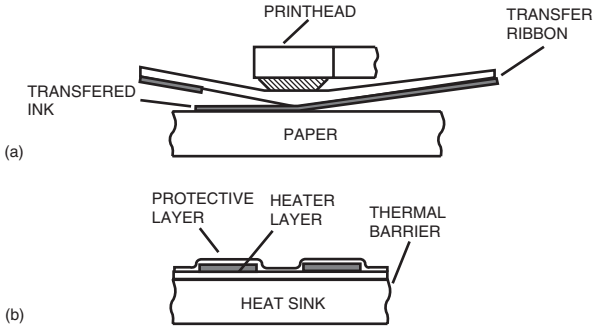


FIGURE 23.11 Schematic of wax transfer process: (a) Intimate contact between printhead, ribbon, and paper is required for successful transfer, (b) design elements of thin-film thermal printhead. Thermal barrier insulates heater for the duration of the heat pulse but allows relation of heater temperature between pulses.

Direct Thermal Transfer

These printers transfer melted wax directly to the paper (Fig. 23.11(a) and Fig. 23.11(b)). The wax that contains the colorant is typically coated at 4 μm thickness onto a polyester film, which, in common implementations, is approximately 6 μm thick. A thermal printhead of the kind described previously presses this ribbon, wax side down, onto the paper. As the individual heating elements are pulsed, the wax melts and transfers by adhesion to the paper. The process is binary in nature; but, by the use of shaped resistors, which produce current crowding via an hourglass shape, for example, the area of wax transferred can be modulated. Common implementations employ page width arrays at 300 dpi with some providing vertical addressability of 600 dpi. The thermal ribbons are also packaged in cassettes for scanning printhead designs in desktop and portable printers. Power consumption is an issue for all thermal printers, and efforts to reduce this for direct thermal transfer have focused on reducing the thickness of the ribbon.

Dye Diffusion Thermal Transfer

This technology involves the transfer of dye from a coated donor ribbon to a receiver sheet via sublimation and diffusion, separately or in combination. The amount of dye transferred is proportional to the amount of heat energy supplied; therefore, this is a continuous tone technology. It has found application as an alternative to silver halide photography, graphics, and prepress proofing. As with all thermal printers the energy is transferred via a transient heating process. This is governed by a diffusion equation and depending on the length of the heating pulse will produce either large temperature gradients over very short distances or lesser gradients extending well outside the perimeter of the resistor. Much of the design, therefore, focuses on the thicknesses of the various layers through which the heat is to be conducted. In the case of thermal dye sublimation transfer a soft-edged dot results, which is suitable for images but not for text. Shorter heating pulses will lead to sharper dots.

Resistive Ribbon Thermal Transfer

This technology is similar to direct thermal transfer in that a thermoplastic ink is imaged via thermal energy onto the substrate. The ribbon is composed of three layers: An electrically conductive substrate of polycarbonate and carbon black (16 μm thick), an aluminum layer 1000–2000 Å, and an ink layer which is typically 5 μm. The aluminum layer serves as a ground return plane. Heat is generated by passing current from an electrode in the printhead in contact with the ribbon substrate through the polycarbonate/carbon layer to the aluminum layer. The high pressure applied through the printhead ensures intimate contact with the paper, which does not have to be especially smooth for successful transfer. Printed characters can be removed by turning on all electrodes at a reduced energy level and heating the ink to the point that it

© 2006 by Taylor & Francis Group, LLC

bonds to the character to be corrected but not to the paper. The unwanted character is removed as the printhead passes over it. This technology does not adapt to color printing in a straightforward way.

23.5 Electrophotographic Printing Technology

Electrophotography is a well established and versatile printing technology. Its first application was in 1960 when it was embodied in an office copier. The process itself bears a strong resemblance to offset lithography. The role of the printing plate is played by a cylindrical drum or belt coated with a photoconductor (PC) on which is formed a printing image consisting of charged and uncharged areas. Depending on the implementation of the technology either the charged or uncharged areas will be inked with a charged, pigmented powder known as toner. The image is offset to the paper either by direct contact or indirectly via a silicone-based transfer drum or belt (similar to the blanket cylinder in offset lithography). Early copiers imaged the material to be copied onto the photoconductor by means of geometrical optics. Replacing this optical system with a scanning laser beam, or a linear array of LEDs, which could be electronically modulated, formed the basis of today's laser printers. As a technology it spans the range from desktop office printers (4–10 ppm) to high-speed commercial printers (exceeding 100 ppm). Although capable of E-size printing its broadest application has been in the range of $8\frac{1}{2}$ in to 17 in wide, in color and in black and white.

Printing Process Steps

Electrophotographic printing involves a sequence of interacting processes which must be optimized collectively if quality printing is to be achieved. With respect to Fig. 23.12 they are as follows.

1. Charging of the photoconductor to achieve a uniform electrostatic surface charge can be done by means of a corona in the form of a thin, partially shielded wire maintained at several kilovolts with respect to ground (corotron). For positive voltages, a positive surface charge results from ionization in the vicinity of the wire. For negative voltages, negative surface charge is produced but by a more complex process involving secondary emission, ion impact, etc., that makes for a less uniform discharge. The precise design of the grounded shield for the corona can have a significant effect on the charge uniformity produced. To limit ozone production, many office printers (<20 ppm) employ a charge roller in contact with the

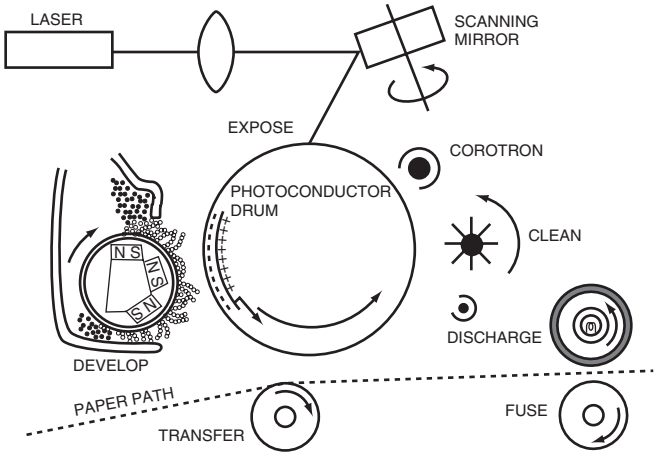


FIGURE 23.12 Schematic of the electrophotographic process. Dual component development is shown with hot roll fusing and coronas for charging and cleaning. (*Source:* Durbeck, R.C. and Sherr, S. 1988. *Hardcopy Output Devices.* Academic Press, San Diego, CA. With permission.)

© 2006 by Taylor & Francis Group, LLC

photoconductor. A localized, smaller discharge occurs in the gap between the roller and photoconductor, reducing ozone production between two and three orders of magnitude.

2. The charged photoconductor is exposed as described previously to form an image that will be at a significant voltage difference with respect to the background. The particular properties of the photoconductor in this step relate to electron hole generation by means of the light and the transport of either electron or hole to the surface to form the image. This process is photographic in nature and has a transfer curve reminiscent of the **H and D** curves for silver halide. The discharge must be swift and as complete as possible to produce a significant difference in voltage between charged and uncharged areas if optimum print quality is to be achieved. Dark decay must be held to a minimum and the PC must be able to sustain repeated voltage cycling without fatigue. In addition to having adequate sensitivity to the dominant wavelength to the exposing light, the PC must also have a wear-resistant surface, be insensitive to fluctuations in temperature and humidity, and release the toner completely to the paper at transfer. It is possible for either the discharged or the charged region to serve as the image to be printed. Widespread practice today, particularly in laser printers, makes use of the discharged area.

Early PCs were sensitive to visible wavelengths and relied on sulfur, selenium, and tellurium alloys. With the use of diode laser scanners, the need for sensitivity in the near infrared has given rise to organic photoconductors (OPC), which in their implementation consist of multiple layers, including a submicron-thick charge generation layer and a charge transport layer in the range of 30 μm thick. This enables the optimization of both processes and is in wide-spread use today. A passivation or *wear layer* is used for OPCs, which are too soft to resist abrasion at the transfer stage. In many desktop devices the photoconductive drum is embodied in a replaceable cartridge containing enough toner for the life of the photoconductor. This provides a level of user servicing similar to that for thermal ink jet printers having replaceable printheads.

3. Image formation is achieved by bringing the exposed photoconductor surface in contact with toner particles, which are themselves charged. Electrostatic attraction attaches these particles to form the image. Once again, uniformity is vital, as well as a ready supply of toner particles to keep pace with the development process. Two methods are in widespread use today: dual component, popular for high-speed printing and monocomponent toners commonly found in desktop printers. Dual component methods employ magnetic toner particles in the 10-μm range and magnetizable carrier beads whose characteristic dimension is around 100 μm. Mechanical agitation of the mixture triboelectrically charges the toner particles, and the combination is made to form threadlike chains by means of imbedded magnets in the development roller. This dense array of threads extending from the development roller is called a magnet brush and is rotated in contact with the charged photoconductor (Fig. 23.10). The toner is then attracted to regions of opposite charge and a sensor-controlled replenishment system is used to maintain the appropriate ratio of toner to carrier beads.

Monocomponent development simplifies this process by not requiring carrier beads, replenishment system, and attendant sensors. A much more compact development system results, and there are two implementations: magnetic and nonmagnetic. Magnetic methods still form a magnetic brush but it consists of toner particles only. A technique of widespread application is to apply an oscillating voltage to a metal sleeve on the development roller. The toner brush is not held in contact with the photoconductor but, rather, a cloud of toner particles is induced by the oscillating voltage as particles detach and reattach depending on the direction of the electric field. Nonmagnetic monocomponent development is equally popular in currently available printers. There are challenges in supplying these toners in charged condition and at rates sufficient to provide uniform development at the required print speed. Their desirability derives from lower cost and inherent greater transparency (for color printing applications) due to the absence of magnetic additives.

One way of circumventing the limitations on particle size and the need for some form of brush technique is to use liquid development. The toner is dispersed in a hydrocarbon-based carrier and is charged by means of the electrical double layer that is produced when the toner is taken into solution. Typically, the liquid toner is brought into contact with the photoconductor via a roller. Upon contact, particle transport mechanisms, such as *electrophoresis*, supply toner to the image regions. Fluid carryout is a major challenge

© 2006 by Taylor & Francis Group, LLC

for these printers. To date this has meant commercial use where complex fluid containment systems can be employed. The technique is capable of competing with offset lithography and has also been used for color proofing.

4. The transfer and fuse stage imposes yet more demands on the toner and photoconductor. The toner must be released to the paper cleanly and then fixed to make a durable image (fusing). The majority of fusing techniques employ heat and pressure, although some commercial systems make use of radiant fusing by means of zenon flash tubes. The toner particles must be melted sufficiently to blend together and form a thin film, which will adhere firmly to the substrate. The viscosity of the melted toner, its surface tension, and particle size influence this process. The design challenge for this process step is to avoid excessive use of heat and to limit the pressure so as to avoid smoothing, that is, calendering and/or curling of the paper. Hot-roll fusing passes the toned paper through a nip formed by a heated elastomer-coated roller in contact with an unheated opposing roller that may or may not have an elastomer composition. Some designs also apply a thin film of silicone oil to the heated roller to aid in release of the melted toner from its surface. There is inevitably some fluid carryout under these conditions, as well as a tendency for the silicone oil to permeate the elastomer and degrade its physical properties. Once again materials innovation plays a major role in electrophotography.

5. The final phase involves removal of any remaining toner from the photoconductor prior to charging and imaging for the next impression. Common techniques involve fiber brushes, magnetic brushes, and scraper blades. Coronas to neutralize any residual charge on the PC or background toner are also typical components of the cleaning process. The toner removed in this step is placed in a waste toner hopper to be discarded. The surface hardness of the PC plays a key role in the efficiency of this step. Successful cleaning is especially important for color laser printers since color contrast can make background scatter particularly visible, for example, magenta toner background in a uniform yellow area.

Dot Microstructure

With respect to image microstructure, the design of the toner material, the development technique, and the properties of the photoconductor play key roles. It is desirable to have toner particles as small as possible and in a tightly grouped distribution about their nominal diameter. Composition of toner is the subject of a vast array of publications and patents. Fundamental goals for toner are a high and consistent charge-to-mass ratio, transparency in the case of color, a tightly grouped distribution, and a minimum, preferably no, wrong-sign particles. The latter are primarily responsible for the undesirable background scatter that degrades the print. Recent developments in toner manufacture seek to control this by means of charge control additives which aid in obtaining the appropriate magnitude of charging and its sign. Grayscale in laser printers is achieved by modulating the pulse width of the diode laser. The shape and steepness of the transfer curve, which relates exposure to developed density, is a function of photoconductor properties, development process, and toner properties. It is possible to produce transfer curves of low or high gradient. For text, a steep gradient curve is desirable, but for images a flatter gradient curve provides more control. Since the stability of the development process is subject to ambient temperature and humidity, the production of a stable grayscale color laser printer without print artifacts is most challenging.

23.6 Magnetographic and Ionographic Technologies

These two technologies are toner based but utilize different addressing and writing media. The photoconductor is replaced by a thin magnetizable medium or hard dielectric layer, such as anodized aluminum, which is used in ionographic printers. Magnetographic printers employ a printhead that produces magnetic flux transitions in the magnetizable media by changing the field direction in the gap between the poles of the printhead. These magnetic transitions are sources of strong field gradient and field strength. Development is accomplished by means of magnetic toner applied via a magnetic brush. The toner particles are magnetized and attracted to the medium by virtue of the strong field gradient. Transfer and fusing

© 2006 by Taylor & Francis Group, LLC

proceed in a similar manner to that of electrophotography. Ionographic printers write onto a dielectric coated drum by means of a printhead containing individual electron sources. The electrons are generated in a miniature cavity by means of air breakdown under the influence of an RF field. The electron beam is focused by a screen electrode, and the cavity functions in a manner similar to that of vacuum tube valves. The role of the plate is played by the dielectric coated metal drum held at ground potential. The charge image is typically developed by monocomponent toner followed by a *transfix*, that is, transfer and fuse operation, often without the influence of heat. Both systems require a cleaning process: mechanical scraping for ionography and magnetic scavenging for magnetography.

23.7 System Issues

Processing and communicating data to control today's printers raises significant system issues in view of the material to be printed. Hardcopy output may contain typography, computer-generated graphics, and natural or synthetic images in both color and black and white. The complexity of this information can require a large amount of processing, either in the host computer or in the printer itself. Applications software programs can communicate with the printer in two ways: via a page description language (PDL), or through a printer command set. The choice is driven by the scope of the printed material. If full page layout with text, graphics, and images is the goal, then PDL communication will be needed. For computer generated graphics a graphical language interface will often suffice. However, many graphics programs also provide PDL output capability. Many options exist and a careful analysis of the intended printed material is necessary to determine if a PDL interface is required.

When processing is done in the host computer, it is the function of the printer driver to convert the outline fonts, graphical objects, and images into a stream of bits to be sent to the printer. Functions that the driver may have to perform include digital halftoning, rescaling, color data transformations, and color appearance adjustments among other image processing operations, all designed to enable the printer to deliver its best print quality. Data compression in the host and decompression in the printer may be used to prevent the print speed being limited by the data rate. Printers that do their own internal data processing contain a hardware formatter board whose properties are often quoted as part of the overall specification for the printer. This is typical for printers with a PDL-based interface. Some of the advantages for this approach include speed of communication with the printer and relieving of the host computer of the processing burden, which can be significant for complex documents.

The increase in complexity of printed documents has emphasized several practical system aspects that relate to user needs: visibility and control of the printed process, font management, quick return to the software application, and printer configuration. The degree of visibility and control in the printing process depends on the choice of application and/or operating environment. Fonts, either outline or bit map, may reside on disk, on computer, or printer read-only memory (ROM). To increase speed, outline fonts in use are rasterized and stored in formatter random-access memory (RAM) or computer RAM. Worst cases exist when outline fonts are retrieved at printing and rasterization occurs on a demand basis. This can result in unacceptably slow printing. If quickness of return to the application is important, printers containing their own formatter are an obvious choice. It is necessary, therefore, to take a system view and evaluate the entire configuration (computer hardware; operating system; application program; interconnect; printer formatter, and its CPU, memory, and font storage) to determine if the user needs will be met.

The need to print color images and complex color shaded graphics has brought issues such as color matching, color appearance, and color print quality to the fore. Color printer configuration now includes choices as to halftoning algorithm, color matching method, and, in some cases, smart processing. The latter refers to customized color processing based on whether the object is a text character, image, or graphic. A further complication arises when input devices and software applications also provide some of these services, and it is possible to have color objects suffer redundant processing before being printed. This can severely degrade the print quality and emphasizes the importance of examining the entire image processing chain and turning off the redundant color processing. Color printer configuration choices focus on a tradeoff between print speed and print quality. Halftoning algorithms that minimize visible texture

© 2006 by Taylor & Francis Group, LLC

and high print quality modes that require overprinting consume more processing time. For color images and graphics, the relationship between the CRT image and hard copy is a matter of choice and taste. For color graphics, it is common practice to sacrifice accuracy of the hue in the interests of colorfulness or **saturation** of the print. In the case of natural images, hue accuracy, particularly for flesh tones, is more important, and a different tradeoff is made. Some software and hardware vendors provide a default configuration that seeks to make the best processing choice based on a knowledge of the content to be printed. If more precise control is desired, some understanding of the color reproduction issues represented by the combination of color input and output devices linked by a PC having a color monitor is required. This is the domain of color management.

Color Management

The fundamental issue to be addressed by color management is that of enabling the three broad classes of color devices (input, display, output) to communicate with each other in a system configuration. The technical issue in this is one of data representation. Each device has an internal representation of color information that is directly related to the nature in which it either represents or records that information. For printers it is typically amounts of cyan, magenta, yellow, and often black (CMY, K) ink; for displays, digital counts of red, green, and blue (RGB); and for many input devices, digitized values of RGB. These internal spaces are called *device spaces* and map out the volume in three-dimensional color space that can be accessed by the device. To communicate between the devices these internal spaces are converted either by analytical models or three-dimensional lookup tables (LUTs) into a device-independent space. Current practice is to use **Commision Internationale d'Eclairage (CIE)** colorimetric spaces, based on the **CIE 1931 standard observer** for this purpose. This enables the device space to be related to coordinates that are derived from measurements on human color perception. These conversions are known as device profiles, and the common device independent color space is referred to as the *profile connection space* (PCS). When this is done it is found that each device accesses a different volume in human color space. For example, a CRT cannot display yellows at the level of saturation available on most color printers. This problem, in addition to issues relating to viewing conditions and the user state of adaptation, makes it necessary to perform a significant amount of color processing if satisfactory results are to be obtained. Solutions to this problem are known as *color management methods* (CMM) (Fig. 23.13). It is the goal of color management systems to coordinate and perform these operations.

The purpose of a color management system is, therefore, to provide the best possible color preference matching, color editing, and color file transfer capabilities with minimal performance and ease of use penalties. Three levels of color-management solutions are common available, point solutions, application solutions, and operating system solutions. Point solutions perform all processing operations in the device driver and fit transparently into the system. If color matching to the CRT is desired, either information as to the make of CRT or visual calibration tools are provided to calibrate the CRT to the driver.

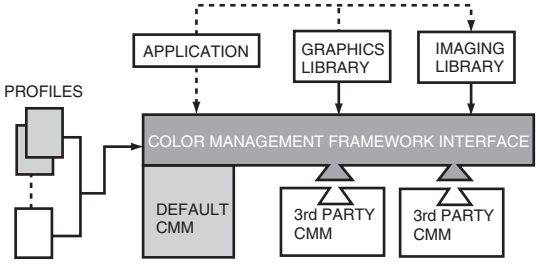


FIGURE 23.13 Proposed ICC color management using ICC profiles. Note fundamental parts: CM framework interface, CMM, third party CMMs, and profiles (which may be resident or embedded in the document).

© 2006 by Taylor & Francis Group, LLC

Application solutions contain libraries of device profiles and associated CMMs. This approach is intended to be transparent to the peripheral and application vendor. Operating system solutions embed the same functionality within the operating system. These systems provide a default color matching method but also allow vendor-specific CMMs to be used.

Although the creation of a device profile involves straightforward measurement processes, there is much to be done if successful color rendition is to be achieved. It is the property of CIE colorimetry that two colors will match when evaluated under the same viewing conditions. It is rarely the case that viewing conditions are identical and it is necessary to perform a number of adaptations commonly called *color appearance transformations* to allow for this. A simple example is to note that the illuminant in a color scanner will have a different color chromaticity than the white point of the CRT, which will also differ from the white point of the ambient viewing illuminant. In addition, as has been mentioned, different devices access different regions of color space; that is, they have different color gamuts. Colors outside the gamut of a destination device such as a printer must therefore be moved to lie within the printer gamut. This will also apply if the dynamic ranges are mismatched between source and destination. Techniques for performing all of the processes are sophisticated and proprietary and reside in vendor specific CMMs.

Defining Terms

Addressability: The spacing of the dots on the page, specified in dots per unit length. This may be different in horizontal and vertical axes and does not imply a given dot diameter.

CIE 1931 standard observer: Set of curves obtained by averaging the results of color matching experiments performed in 1931 for noncolor defective observers. The relative luminances of the colors of the spectrum were matched by mixtures of three spectral stimuli. The curves are often called *color matching curves*.

Commision Internationale d'Eclairage (CIE): International standards body for lighting and color measurement. Central Bureau of the CIE, a-1033 Vienna, P.O. Box 169, Austria.

Digital halftone: Halftone technique based on patterns of same size dots designed to simulate a shade of gray between white paper and full colorant coverage.

Grayscale: Intrinsic modulation property of the marking technology that enables either dots of different size or intensity to be printed.

Halftone: Technique of simulating continuous tones by varying the amount of area covered by the colorant. Typically accomplished by varying the size of the printed dots in relation to the desired intensity.

H and D curve: Characteristic response curve for a photosensitive material that relates exposure to produced/developed optical density.

Resolution: Spacing of the printer dots such that full ink coverage is just obtained. Calculated from the dot size and represents the fundamental ability of the printer to render fine detail.

Saturation: When applied to color it describes the colorfulness with respect to the achromatic axis. A color is saturated to the degree that it has no achromatic component.

References

Cornsweet, T.N. 1970. *Visual Perception*. Academic Press, New York.

Diamond, A.S., ed. 1991. *Handbook of Imaging Materials*. Marcel Dekker, New York.

Durbeck, R.C. and Sherr, S. 1988. *Hardcopy Output Devices*. Academic Press, San Diego, CA.

Hunt, R.W.G. 1992. *Measuring Color,* 2nd ed. Ellis Horwood, England.

Hunt, R.W.G. 1995. *The Reproduction of Color,* 5th ed. Fountain Press. England.

Scharfe, M. 1984. *Electrophotography Principles and Optimization*. Research Studies Press Ltd., Letchworth, Hertfordshire, England.

Schein, L.B. 1992. *Electrophotography and Development Physics,* 2nd ed. Springer–Verlag, Berlin.

© 2006 by Taylor & Francis Group, LLC

Schreiber, W.F. 1991. *Fundamentals of Electronic Imaging Systems,* 2nd ed. Springer–Verlag, Berlin.
Ulichney, R. 1987. *Digital Halftoning.* MIT Press, Cambridge, MA.
Williams, E.M. 1984. *The Physics and Technology of Xerographic Processes.* Wiley-Interscience, New York.

Further Information

Color Business Report: published by Blackstone Research Associates, P.O. Box 345, Uxbridge, MA 01569-0345. Covers industry issues relating to color, computers, and reprographics.

International Color Consortium: The founding members of this consortium include Adobe Systems Inc., Agfa-Gevaert N.V., Apple Computer, Inc., Eastman Kodak Company, FOGRA (Honorary), Microsoft Corporation, *Hewlett-Packard Journal,* 1985. 36(5); 1988. 39(4) (Entire issues devoted to Thermal Ink Jet).

Journal of Electronic Imaging: co-published by IS&T and SPIE. Publishes papers on the acquisition, display, communication and storage of image data, hardcopy output, image visualization, and related image topics. Source of current research on color processing and digital halftoning for computer printers.

Journal of Imaging Science and Technology: official publication of IS&T, which publishes papers covering a broad range of imaging topics, from silver halide to computer printing technology.

The International Society for Optical Engineering, SPIE, P.O. Box 10, Bellingham, Washington 98227-0010, sponsors conferences in conjunction with IS&T on electronic imaging and publishes topical proceedings from the conference sessions.

The *Hardcopy Observer,* published monthly by Lyra Research Services, P.O. Box 9143, Newtonville, MA 02160. An industry watch magazine providing an overview of the printer industry with a focus on the home and office.

The Society for Imaging Science and Technology, IS&T, 7003 Kilworth Lane, Springfield, VA 22151. Phone (703)642 9090, Fax (703)642 9094. Sponsors wide range of technical conferences on imaging and printing technologies. Publishes conference proceedings, books, *Journal of Electronic Imaging, Journal of Imaging Science and Technology, IS&T Reporter*.

The Society for Information Display, 1526 Brookhollow Drive, Ste 82, Santa Ana, CA 92705-5421, Phone (714)545 1526, Fax (714)545 1547. Cosponsors annual conference on color imaging with IS&T.

© 2006 by Taylor & Francis Group, LLC

24

Data Storage Systems

24.1 Introduction 24-1
24.2 Redundant Arrays of Independent Disks (RAID) Systems 24-1
RAID Elements • RAID Levels

Jerry C. Whitaker

24.1 Introduction

The astounding volume of data being transmitted between systems today has created an obvious need for data management. As a result, more servers—whether they are PCs, UNIX workstations, minicomputers, or supercomputers—have assumed the role of information providers and managers. The number of networked or connectable systems is increasing by leaps and bounds as well, thanks to the widespread adoption of the client-server computing model. Hard disk storage plays an important role in enabling improvements to networked systems, because the vast and growing ocean of data needs to reside somewhere. It also has to be readily accessible, placing a demand upon storage system manufacturers to not only provide high-capacity products, but products that can access data as fast as possible and to as many people at the same time as possible. Such storage also needs to be secure, placing an importance on reliability features that best ensure data will never be lost or otherwise rendered inaccessible to network system users.

24.2 Redundant Arrays of Independent Disks (RAID) Systems

The common solution to providing access to many gigabytes of data to users fast and reliably has been to assemble a number of drives together in a gang or array of disks, known as **redundant arrays of independent disks (RAID)** subsystems. Simple RAID subsystems are basically a cluster of up to five or six disk drives assembled in a cabinet that are all connected to a single controller board. The RAID controller orchestrates read and write activities in the same way a controller for a single disk drive does, and treats the array as if it were in fact a single or **virtual drive**. RAID management software that resides in the host system provides the means to manage data to be stored on the RAID subsystem. A typical RAID configuration is illustrated in Fig. 24.1.

RAID Elements

Despite its multidrive configuration, the individual disk drives of a RAID subsystem remain hidden from users; the subsystem itself is the virtual drive, though it can be infinitely large. The phantom virtual drive is created at a lower level within the host operating system through the RAID management software. Not only does the software set up the system to address the RAID unit as if it were a single drive, it allows the subsystem to be configured in ways that best suit the general needs of the host system.

RAID subsystems can be optimized for performance, highest capacity, fault tolerance, or a combination of these attributes. Different **RAID levels** have been defined and standardized in accordance with these

© 2006 by Taylor & Francis Group, LLC

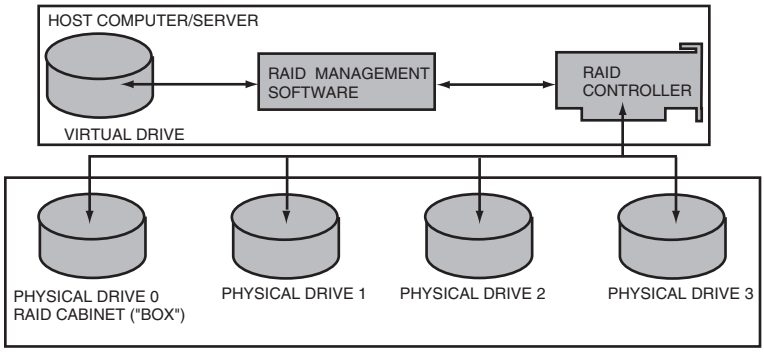


FIGURE 24.1 A typical RAID configuration. (*Source:* Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.)

general optimization parameters. There are six such standardized levels RAID, called RAID 0, 1, 2, 3, 4, and 5, depending on performance, redundancy and other attributes required by the host system.

The RAID controller board is the hardware element that serves as the backbone for the array of disks; it not only relays the input/output (I/O) commands to specific drives in the array, but provides the physical link to each of the independent drives so that they may easily be removed or replaced. The controller also serves to monitor the integrity of each drive in the array to anticipate the need to move data should it be placed in jeopardy by a faulty or failing disk drive (a feature known as *fault tolerance*).

RAID Levels

The RAID 0–5 standards offer users and system administrators a host of configuration options. These options permit the arrays to be tailored to their application environments. Each of the various configurations focus on maximizing the abilities of an array in one or more of the following areas:

- Capacity
- Data availability
- Performance
- Fault tolerance

RAID Level 0

An array configured to RAID level 0 is an array optimized for performance, but at the expense of fault tolerance or data integrity. RAID level 0 is achieved through a method known as **striping**. The collection of drives (*virtual drive*) in a RAID level 0 array has data laid down in such a way that it is organized in stripes across the multiple drives. A typical array can contain any number of stripes, usually in multiples of the number of drives present in the array. Take, as an example, a four-drive array configured with 12 stripes (four stripes of designated space per drive). Stripes 0, 1, 2, and 3 would be located on corresponding hard drives 0, 1, 2, and 3. Stripe 4, however, appears on a segment of drive 0 in a different location than stripe 0; stripes 5–7 appear accordingly on drives 1, 2, and 3. The remaining four stripes are allocated in the same even fashion across the same drives such that data would be organized in the manner depicted in Fig. 24.2. Practically any number of stripes can be created on a given RAID subsystem for any number of drives: 200 stripes on two disk drives is just as feasible as 50 stripes across 50 hard drives. Most RAID subsystems, however, tend to have between 3 and 10 stripes.

The reason RAID 0 is a performance-enhancing configuration is that striping enables the array to access data from multiple drives at the same time. In other words, because the data is spread out across a number of drives in the array, it can be accessed faster because it is not bottled up on a single drive. This is especially beneficial for retrieving a very large file, because it can be spread out effectively across multiple drives and accessed as if it were the size of any of the fragments it is organized into on the data stripes.

© 2006 by Taylor & Francis Group, LLC

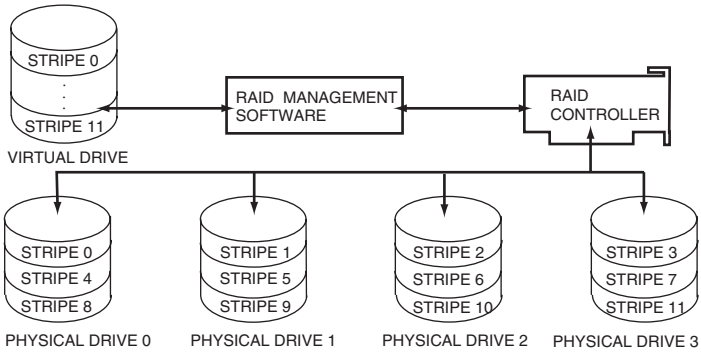


FIGURE 24.2 In a RAID level 0 configuration, a virtual drive comprises several stripes of information. Each consecutive stripe is located on the next drive in the chain, evenly distributed over the number of drives in the array. (*Source:* Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.)

The downside to RAID level 0 configurations is that it sacrifices fault tolerance, raising the risk of data loss because no room is made available to store redundant data. If one of the drives in the RAID 0 fails for any reason, there is no way of retrieving the lost data, as can be done in other RAID implementations.

RAID Level 1

The RAID level 1 configuration employs what is known as *disk mirroring*, which is done to ensure data reliability or a high degree of fault tolerance. RAID 1 also enhances read performance, but the improved performance and fault tolerance come at the expense of available capacity in the drives used. In a RAID level 1 configuration, the RAID management software instructs the subsystem's controller to store data redundantly across a number of the drives (mirrored set) in the array. In other words, the same data is copied and stored on different disks to ensure that, should a drive fail, the data is available somewhere else within the array. In fact, all but one of the drives in a mirrored set could fail and the data stored to the RAID 1 subsystem would remain intact. A RAID level 1 configuration can consist of multiple mirrored sets, whereby each mirrored set can be a different capacity. Usually the drives making up a mirrored set are of the same capacity. If drives within a mirrored set are of different capacities, the capacity of a mirrored set within the RAID 1 subsystem is limited to the capacity of the smallest capacity drive in the set, hence the sacrifice of available capacity across multiple drives.

The read performance gain can be realized if the redundant data is distributed evenly on all of the drives of a mirrored set within the subsystem. The number of read requests and total wait state times both drop significantly, inversely proportional to the number of hard drives in the RAID, in fact. To illustrate, suppose three read requests are made to the RAID level 1 subsystem (see Fig. 24.3). The first request looks for data in the first block of the virtual drive; the second request goes to block 0, and the third seeks from block 2. The host-resident RAID management software can assign each read request to an individual drive. Each request is then sent to the various drives, and now—rather than having to handle the flow of each data stream one at a time—the controller can send three data streams almost simultaneously, which in turn reduces system overhead.

RAID Level 2

RAID level 2 is rarely used in commercial applications, but is another means of ensuring data is protected in the event drives in the subsystem incur problems or otherwise fail. This level builds fault tolerance around Hamming error correction code (ECC), which is often used in modems and solid-state memory devices as a means of maintaining data integrity. ECC tabulates the numerical values of data stored on specific blocks in the virtual drive using a formula that yields a checksum. The checksum is then appended to the end of the data block for verification of data integrity when needed.

© 2006 by Taylor & Francis Group, LLC

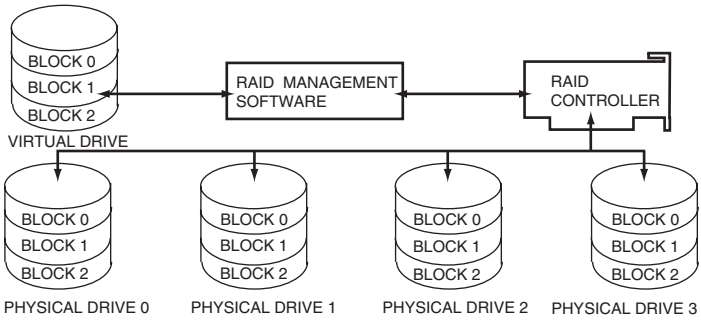


FIGURE 24.3 A RAID level 1 subsystem provides high data reliability by replicating (*mirroring*) data between physical hard drives. In addition, I/O performance is boosted as the RAID management software allocates simultaneous read requests between several drives. (*Source:* Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.)

As data is read back from the drive, ECC tabulations are again computed, and specific data block checksums are read and compared against the most recent tabulations. If the numbers match, the data is intact; if there is a discrepancy, the lost data can be recalculated using the first or earlier checksum as a reference point, as illustrated in Table 24.1.

This form of ECC is actually different from the ECC technologies employed within the drives themselves. The topological formats for storing data in a RAID level 2 array is somewhat limited, however, compared to the capabilities of other RAID implementations, which is why it is not commonly used in commercial applications.

RAID Level 3

This RAID level is essentially an adaptation of RAID level 0 that sacrifices some capacity, for the same number of drives, but achieves a high level of data integrity or fault tolerance. It takes advantage of RAID level 0 data striping methods, except that data is striped across all but one of the drives in the array. This drive is used to store parity information for maintenance of data integrity across all drives in the subsystem. The parity drive itself is divided into stripes, and each parity drive stripe is used to store parity information for the corresponding data stripes dispersed throughout the array. This method achieves high data transfer performance by reading from or writing to all of the drives in parallel or simultaneously but retains the means to reconstruct data if a given drive fails, maintaining data integrity for the system. This concept is illustrated in Fig. 24.4. RAID level 3 is an excellent configuration for moving very large sequential files in a timely manner.

The stripes of parity information stored on the dedicated drive are calculated using the Exclusive OR function. By using Exclusive OR with a series of data stripes in the RAID, any lost data can easily be recovered. Should a drive in the array fail, the missing information can be determined in a manner similar to solving for a single variable in an equation.

TABLE 24.1 Example of Error Correction Coding

Assume the phrase being stored is **HELLOTHERE**. The checksum is computed for every 10 bytes of data.

Data being stored:	**H**	**E**	**L**	**L**	**O**	**T**	**H**	**E**	**R**	**E**
Numerical representation:	72	69	76	76	79	84	72	69	82	69
Checksum formula:	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$	$x7$	$x8$	$x9$	$x10$
Multiplied out:	72	138	228	304	395	504	504	414	738	690
[Check]sum of all values:	72	+138	+228	+304	+395	+504	+504	+414	+738	+690
										=3987

Thus, the data is stored on the drive as: 72 69 76 76 79 84 72 69 82 69 3987
As the data is read back from the drive, the same calculations with the data segment are made. The newly computed checksum is compared against the previously stored checksum, thus verifying data integrity.

Source: Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.

© 2006 by Taylor & Francis Group, LLC

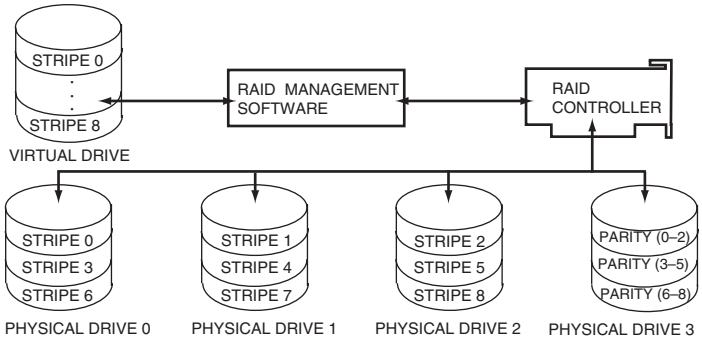


FIGURE 24.4 A RAID level 3 configuration is similar to a RAID level 0 in its utilization of data stripes dispersed over a series of hard drives to store data. In addition to these data stripes, a specific drive is configured to hold parity information for the purpose of maintaining data integrity throughout the RAID subsystem. (*Source:* Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.)

RAID Level 4

This level of RAID is similar in concept to RAID level 3, but emphasizes performance for different applications, e.g., database files vs large sequential files. Another difference between the two is that RAID level 4 has a larger stripe depth, usually of two blocks, which allows the RAID management software to operate the disks more independently than RAID level 3 (which controls the disks in unison). This essentially replaces the high data throughput capability of RAID level 3 with faster data access in read-intensive applications. (See Fig. 24.5.)

A shortcoming of RAID level 4 is rooted in an inherent bottleneck on the parity drive. As data is written to the array, the parity encoding scheme tends to be more tedious in write activities than with other RAID topologies. This more or less relegates RAID level 4 to read-intensive applications with little need for similar write performance. As a consequence, like level 3, level 4 does not see much common use in commercial applications.

RAID Level 5

This is the last of the most common RAID levels in use, and is probably the most frequently implemented. RAID level 5 minimizes the write bottlenecks of RAID level 4 by distributing parity stripes over a series of

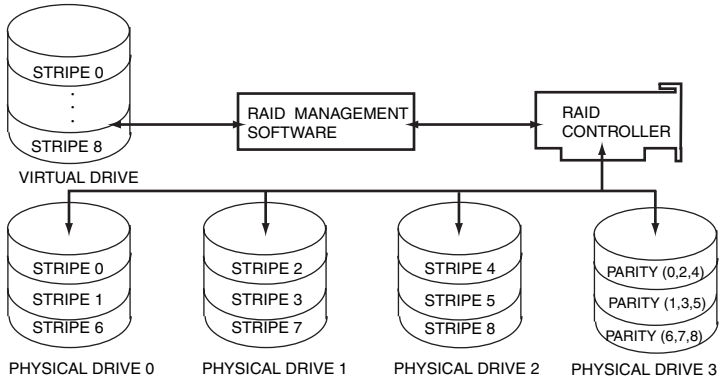


FIGURE 24.5 RAID level 4 builds on RAID level 3 technology by configuring parity stripes to store data stripes in a nonconsecutive fashion. This enables independent disk management, ideal for multiple-read-intensive environments. (*Source:* Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.)

© 2006 by Taylor & Francis Group, LLC

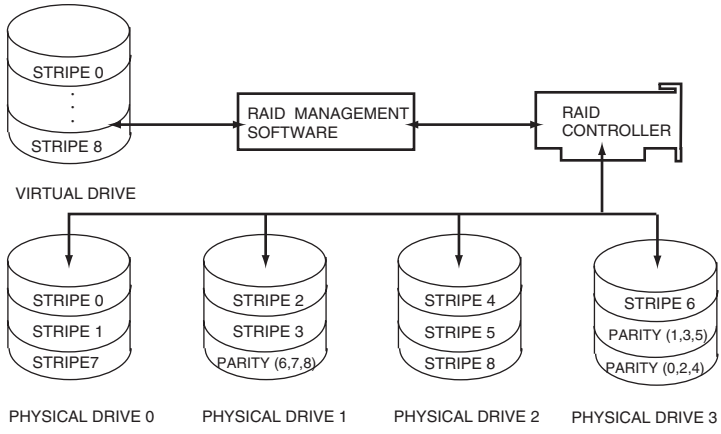


FIGURE 24.6 RAID level 5 overcomes the RAID level 4 write bottleneck by distributing parity stripes over two or more drives within the system. This better allocates write activity over the RAID drive members, thus enhancing system performance. (*Source:* Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.)

hard drives. In so doing it provides relief to the concentration of write activity on a single drive, which in turn enhances overall system performance. (See Fig. 24.6.)

The way RAID level 5 reduces parity write bottlenecks is relatively simple. Instead of allowing any one drive in the array to assume the risk of a bottleneck, all of the drives in the array assume write activity responsibilities. This distribution frees up the concentration on a single drive, improving overall subsystem throughput. The RAID level 5 parity encoding scheme is the same as levels 3 and 4, and maintains the system's ability to recover lost data should a single drive fail. This can happen as long as no parity stripe on an individual drive stores the information of a data stripe on the same drive. In other words, the parity information for any data stripe must always be located on a drive other than the one on which the data resides.

Other RAID Levels

Other, less common RAID levels have been developed as custom solutions by independent vendors. Those levels include

- RAID level 6, which emphasizes ultrahigh data integrity
- RAID level 10, which focuses on high I/O performance and very high data integrity
- RAID level 53, which combines RAID levels 0 and 3 for uniform read and write performance

TABLE 24.2 Summary of RAID Level Properties

RAID Level	Capacity	Data Availability	Data Throughput	Data Integrity
0	High	Read/write high	High I/O transfer rate	
1		Read/write high		Mirrored
2	High		High I/O transfer rate	ECC
3	High		High I/O transfer rate	Parity
4	High	Read high		Parity
5	High	Read/write high		Parity
6		Read/write high		Double parity
10		Read/write high	High I/O transfer rate	Mirrored
53			High I/O transfer rate	Parity

Source: Adapted from Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.

© 2006 by Taylor & Francis Group, LLC

Custom RAID Systems

Perhaps the greatest advantage of RAID technology is the sheer number of possible adaptations available to users and systems designers. RAID offers the ability to customize an array subsystem to the requirements of its environment and the applications demanded of it. The inherent variety of configuration options provides several ways in which to satisfy specific application requirements, as detailed in Table 24.2. Customization, however, does not stop with a RAID level. Drive models, capacities, and performance levels have to be factored in as well as what connectivity options are available.

Defining Terms

Fiber channel-arbitrated loop (FC-AL): A high-speed interface protocol permitting high data transfer rates, large numbers of attached devices, and long-distance runs to remote devices using a combination of fiber optic and copper components.

Redundant arrays of independent disks (RAID): A configuration of hard disk drives and supporting software for mass storage whose primary properties are high capacity, high speed, and reliability.

RAID level: A standardized configuration of RAID elements, the purpose of which is to achieve a given objective, such as highest reliability or greatest speed.

Single connector attachment (SCA): A cabling convention for certain interface standards that simplifies the interconnection of devices by combining data and power signals as a common, standardized port.

Small computer systems interface (SCSI): A cabling and software protocol used to interface multiple devices to a computer system. These devices may be internal and/or external to the computer itself. Several variations of SCSI exist.

Striping: A hard disk storage organization technique whereby data is stored on multiple physical devices so as to increase write and read speed.

Thermal calibration: A housekeeping function of a hard disk drive used to maintain proper alignment of the head with the disk surface.

Virtual drive: An operational state for a computing device whereby memory is organized to achieve a specific objective that usually does not exist as an actual physical entity. For example, RAM in the computer may be made to appear as a physical drive, or two or more hard disks can be made to appear as a single physical drive.

References

Anderson, D. 1995. Fiber channel-arbitrated loop: The preferred path to higher I/O performance, flexibility in design. Seagate Technology Paper No MN-24, Seagate, Scotts Valley, CA.

Heyn, T. 1995. The RAID Advantage. Seagate Technology Paper, Seagate, Scotts Valley, CA.

Tyson, H. 1995. Barracuda and elite: Disk drive storage for professional audio/video. Seagate Technology Paper No. SV-25, Seagate, Scotts Valley, CA.

Further Information

The technology behind designing and manufacturing hard disk drives is beyond the scope of this chapter, and because most of the applied use of disks involves treating the drive as an operational component (black box, if you will), the best source of current information is usually drive manufacturers. Technical application notes and detailed product specifications are typically available at little or no cost.

© 2006 by Taylor & Francis Group, LLC

25

Optical Storage Systems

25.1 Introduction 25-1
25.2 The Optical Head 25-1
The Servosystem • Optical Recording and Read Channel • Phase-Change Recording
25.3 Worm Technology 25-8
25.4 Magneto-Optic Technology 25-8
25.5 Compact Disk-Recordable (CD-R) 25-10
Recording Modes of CD-R
25.6 Optical Disk Systems 25-12
Disks • Automated Optical Storage Systems • Future Technology and Improvements in Optical Storage

Praveen Asthana

25.1 Introduction

Recordable optical disk drive technology provides a well-matched solution to the increasing demands for removable storage. An optical disk drive provides, in a sense, infinite storage capabilities: Extra storage space is easily acquired by using additional media cartridges (which are relatively inexpensive). Such cost effective storage capabilities are welcome in storage-intensive modern computer applications such as desktop publishing, computer aided design/computer aided manufacturing (CAD/CAM), or multimedia authoring.

25.2 The Optical Head

The purposes of the optical head are to transmit the laser beam to the optical disk, focus the laser beam to a diffraction limited spot, and to transmit readout signal information from the optical disk to the data and servo-detectors.

The laser diode is a key component in optical storage, whether the recording technology is magneto-optic, ablative WORM, or phase change. Early generations of optical drives used infrared lasers emitting in the 780 nm or 830 nm wavelengths. Later generation of drives use red laser wavelengths emitting at around 690 nm. The lasers are typically rated to have a maximum continuous output power in the 40-mW range and are index guided to ensure good wavefront quality.

In a laser diode, light is emitted from the *facets*, which are the cleaved ends of the waveguide region of an index guided laser. The facet dimensions are small enough (order of a few micrometers) for diffraction to take place as light is emitted from the facet. As a result the output beam has a significant divergence angle. In many commercial laser diodes, the width of the facet (i.e., the dimensions parallel to the pn-junction plane) is much larger than the height (or the direction perpendicular to the junction plane), which causes

© 2006 by Taylor & Francis Group, LLC

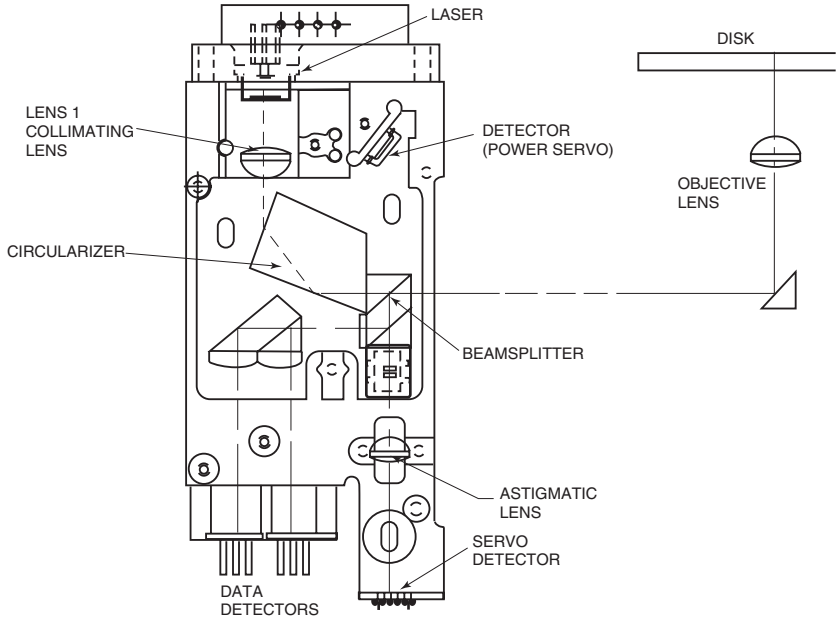


FIGURE 25.1 Drawing of the optical head in a typical magneto-optic disk drive. Shown is a split-optics design, which consists of a fixed set of components (the laser, detectors, and polarizing optics) and a movable system consisting of a beam bender and an objective lens that focuses the light on the disk. (*Source:* Asthana, P. 1994. Laser Focus World, Jan., p. 75. Penwell Publishing Co., Nashua, N.H. Used by permission.)

the divergence angles to be unequal in the directions parallel and perpendicular to the laser junction. The spatial profile of the laser beam some distance from the laser is thus elliptical.

The basic layout of the optical head of a magneto-optic drive is shown in Fig. 25.1. The laser is mounted in a heat sink designed to achieve an athermal response. The output from the laser diode is collimated by lens 1. A prismlike optical element called a circularizer is then used to reduce the ellipticity of the laser beam. The beam then passes through a polarizing beamsplitter, which reflects part (30%) of the beam toward a detector and transmits the rest toward the disk.

The output of the laser is linearly polarized in the direction parallel to the junction (which will be referred to as P-polarization). The ratio of the intensity of the P-polarization component of the emitted light to the ratio of the intensity in the S-polarization component is > 25:1. This polarizing beam splitter is designed to transmit 70% of the P-polarized light and 100% of the S-polarized light. The light that is reflected is incident on a light detector, which is part of a power servoloop designed to keep the laser at a constant power. Without a power servoloop, the laser power will fluctuate with time as the laser junction heats up, which can adversely affect the read performance.

The beam transmitted by the beam splitter travels to a turning (90°) mirror, called a *beam bender*, which is mounted on a movable actuator. During track seeking operations, this actuator can move radially across the disk. The beam reflected by the turning mirror is incident on an objective lens (also mounted on the actuator), which focuses the light on the disk. This type of optical head design in which the laser, the detectors, and most of the optical components are stationary while the objective lens and beambender are movable is called a *split optics design*. In early optical drive designs, the entire optical head was mounted on a actuator and moved during seeking operations. This led to slow **seek times** (∼200 ms) because of the mass on the actuator. A split optics design, which is possible because coherent light can be made highly collimated, lowers the mass on the actuator and thus allows much faster seek times.

The size of the focal spot formed by the objective lens on the disk depends on the *numerical aperture* (NA) of the lens and the amount of *overfill* of the lens (i.e., the amount with which the diameter of the incident collimated beam exceeds the aperture of the objective lens). The numerical aperture is given by

© 2006 by Taylor & Francis Group, LLC

$NA = n \sin\theta_{max}$, in which n is the refractive index of the lens and θ_{max} is the incidence angle of a light ray focused through the margin of the lens. The beam of light incident on the objective lens usually has a Gaussian electric field (or intensity) profile. The profile of the focused spot is a convolution of the incident intensity profile and the aperture of the objective lens (Goodman, 1968). Overfilling the lens aperture reduces the size of the focused spot at the cost of losing optical energy outside the aperture and increasing the size of the side lobes. Optimization of the amount of overfill (Marchant, 1990) yields an approximate spot diameter of

$$D = 1.18\lambda/NA \tag{25.1}$$

in which λ is the wavelength of light and NA is the numerical aperture of the objective lens. The depth of focus z of the focal spot is given by $z = 0.8\lambda/(NA)^2$.

The depth of focus defines the accuracy with which the objective lens position must be held with respect to the disk surface. The smaller the depth of focus, the less tolerance the system has for media tilt and the more difficult the job of the focus servosystem. Thus trying to reduce the spot size (always a goal as a smaller spot allows a higher storage density) by increasing the *NA* of the lens becomes impractical beyond an *NA* of about 0.6. The objective lens also acts as a collector lens for the light that is reflected from the disk. This reflected light, used for the servosystems and during reading, contains the readout information. The reflected light follows the incident path up to the fixed optical element. Beamsplitter 1 reflects all of the S polarized light and 30% of the P polarized light in the direction of the servo and data detectors.

The portion of the light that is transmitted by the beamsplitter is, unfortunately, focused by the collimating lens back into the facet of the laser. This feedback light causes a number of problems in the laser, even though the net amount of feedback does not exceed about 7% of the output light for most magneto-optic media. Optical feedback affects the laser by causing the laser to mode hop randomly, which results in a random amplitude fluctuation in the output. This amplitude noise can be a serious problem and so techniques must be used to control the laser noise (such as injection of high-frequency current) or HFM (Arimoto, 1986). Increasing the HFM current in general can decrease the amount of noise, but as a practical matter, the injection current cannot be made arbitrarily large as it may then violate limits on allowable radiation from computer accessories. Optical feedback also decreases the threshold and increases the slope of the power-current (or PI) curve of the laser, but these effects are not really a problem.

The light reflected by beamsplitter 1 is further split by beamsplitter 2 into servo and data components. The light reflected by the beamsplitter is incident on the data detectors. For magneto-optic read back, two detectors are used in a technique known as *differential detection* (i.e., the difference in the signals incident on the two detectors is taken). The light transmitted through beamsplitter 2 is incident onto a special multielement servodetector and is used to generate the servosignals. The mechanism by which these signals are generated are the subject of the following discussion on the **servosystem**.

The Servosystem

The servosystem is what enables the focused laser spot to be positioned with accuracy onto any of the tracks on the disk and ensure that it can be moved to any track on the disk as required. The high track densities (18,000 tracks/in) on optical disks require that the laser spot position be controlled to within a fraction of a micrometer. To be able to move across the entire disk surface requires a large actuator, but such an actuator will be too massive to quickly respond to the rapid changes in the track position (due to run-out in the disk) as the disk spins. Therefore, a compound actuator consisting of a coarse actuator and a fine actuator is used to control the radial position of the laser beam on the disk. The fine actuator, which has a very low mass, can change the spot position rapidly over a limited range. The coarse actuator has a slower response, but has a much wider range of motion and is used for long seek operations. Optical disks have a continuous spiral groove (as in a phonograph record) to provide information on the relative track location.

In addition to **tracking** and seeking, the laser spot in an optical drive must be kept in perfect focus on the disk regardless of the motion of the disk (there can be quite a lot of vertical motion if the disk has tilt

© 2006 by Taylor & Francis Group, LLC

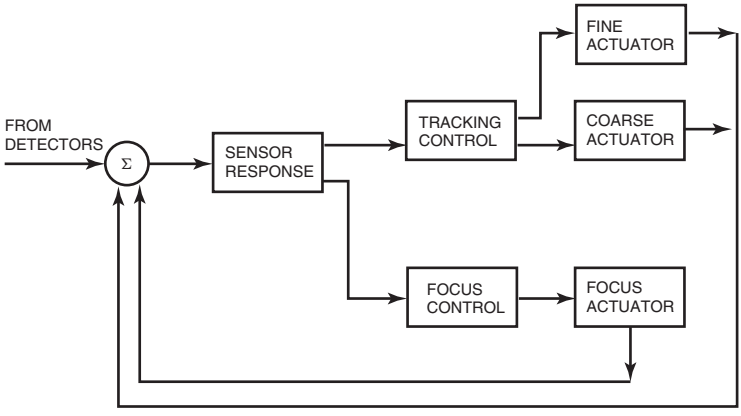


FIGURE 25.2 Block diagram of the servocontrol system for an optical disk drive.

or is slightly warped). To do this, the objective lens must be constantly adjusted to correct for the axial motion of the disk surface as the media spins. The lens position is controlled by a focus servomechanism.

Figure 25.2 shows a block diagram of an optical drive servocontrol system (combined tracking and focusing). The return beam contains information on the focus and position of the spot, which is processed by the servo detectors. The feedback signals derived from the detectors allow the system to maintain control of the beam.

The focus control system requires a feedback signal that accurately indicates the degree and direction of focus error (Braat and Bouwhuis, 1978; Earman, 1982). To generate a focus error signal, an astigmatic lens is used to focus onto a quadrant detector, a portion of the light reflected from the disk. In perfect focus, the focal spot is equally distributed on the four elements of the quad detector (as shown in Fig. 25.3). However, if the lens is not in focus, the focal spot on the detector is elliptical because of the optical properties of the astigmatic lens. The unequal distribution of light on the detector quadrants generates a *focus error signal* (FES). This signal is normalized with respect to light level to make it independent of laser power and disk reflectivity. The focus actuator typically consists of an objective lens positioned by a small linear voice coil motor. The coils are preferably mounted with the lens to reduce moving mass, while the permanent magnets are stationary. The lens can be supported by either a bobbin on a sliding pin or elastic flexures. The critical factors in the design are range of motion, acceleration, freedom from resonances, and thermal considerations.

Once the spot is focused on the active surface, it must find and maintain position along the desired track. This is the role of the tracking servo. The same quadrant detector that is used to generate the focus signal can be used to generate the *tracking error signal* (TES). The beam returning from the disk contains first-order diffraction components; their intensity depends on the position of the spot on the tracks and varies the light falling along one axis of the quadrant detector. The TES is the normalized difference of the current from the two halves of the detector, and it peaks when the spot passes over the cliff between a land and groove (Mansuripur, 1987; Braat and Bouwhuis, 1978).

As a matter of terminology, the **seek time** usually refers to the actual move time of the actuator. The time to get to data, however, is called the **access time** which includes the latency of the spinning disk in addition

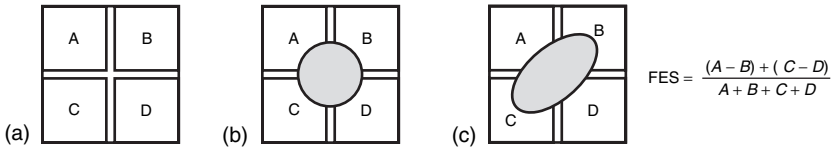


FIGURE 25.3 Focus control system: (a) the quad detector, (b) the spot of light focused on the quad by the astigmatic lens (circular implies objective lens is in focus), and (c) the spot of light on the quad when the objective lens is out of focus.

© 2006 by Taylor & Francis Group, LLC

to the seek time. For example, a drive spinning a disk at 3600 rpm has a latency of $(0.5 \times (60/3600))$ s or 8 ms. Thus, a 3600-rpm drive with a seek time of 30 ms will have an access time of 38 ms. The standard way to measure seek time is 1/3 of the full stroke of the actuator (i.e., the time it takes to cover 1/3 of the disk). This is a historical artifact from the times of early hard disk drives.

The ability to accurately follow the radial and axial motions of the spinning disk results directly from the quality of the focus and tracking actuators. To reject the errors due to shock, vibration, and media runout, the servosystem must have high bandwidth. The limitation to achieving high bandwidth is usually the resonance modes of the actuator. As the actuators are reduced in size, the frequencies of the resonances become higher and the achievable bandwidth of the system rises. Servosystems in optical drives face additional challenges because they have to handle removable media, which has variations between different disks (such as media tilt).

Optical Recording and Read Channel

A schematic block diagram of the functions in an optical drive is shown in Fig. 25.4. The SCSI controller handles the flow of information to and from the host (including commands). The optical disk controller is a key controller of the data path. It interprets the commands from the SCSI controller and channels data appropriately through the buffer random access memory (RAM) to the write channel, or from the read channel to the output. The drive control microprocessor unit controls, through the logic gate arrays, all the functions of the optical drive including the servo control, spindle motor, actuators, laser driver, etc.

Data that are input to the drive over the SCSI for recording are first broken up into fixed **block sizes** (of, for example, 512 kilobyte or 1024 kilobytes length) and then stored in the data buffer RAM. Magneto-optic, phase-change, and WORM drives can be classified as *fixed block architecture* technologies in which data blocks are recorded much like in hard drives (Marchant, 1990). Blocks of data can be placed anywhere on the disk in any sequence. The current CD recordable drive is, on the other hand, an example of a non-fixed block architecture (because its roots are in CD audio). In common CD-R drives, input data are recorded sequentially (like a tape player) and can be of any continuous length.

Error correction and control (**ECC**) bytes are added to each block of data. Optical drives use Reed–Solomon codes, which are able to reduce the error rate from 1E-5 to about 1E-13 (Golomb, 1986). After the addition of ECC information, the data are encoded with run length limited (RLL) modulation codes (Tarzaiski, 1983; Treves and Bloomberg, 1986) in order to increase efficiency and improve detection. Special characters are inserted in the bit stream such as a synchronization character to tell where the data begins. All in all, the overhead required to store customer data is about 20%.

In most current optical drives, recording is based on **pulse position modulation** (**PPM**) techniques. In basic PPM recording, the presence of a mark signifies a one and the absence signifies a zero bit. Thus, a 1001 bit sequence would be recorded as mark-space-space-mark. In **pulse width modulation** (**PWM**) recording, the edges of the mark represent the one bits (Sukeda et al., 1987). Thus a 1001 bit sequence would be recorded as a single mark of a certain size. A longer bit sequence such as 100001 would be represented by a longer mark, hence the term pulse *width* modulation. The use of pulse width modulation allows a higher linear density of recording than pulse position modulation.

A comparison of PWM recording with the current technique of PPM is shown in Fig. 25.5. It is more difficult to implement PWM technology (a more complicated channel is required), and PWM writing has greater sensitivity to thermal effects. Thus implementing PWM is a challenging task for drive vendors.

On readback, the light reflects from the disk and travels to a photodetector (or two detectors in the case of magneto-optic drives). For WORM, phase-change, and CD-R disks, the signal is intensity modulated and thus can be converted to current from the detectors and amplified prior to processing. The WORM, phase-change, and CD-R disks have a high contrast ratio between mark and no-mark on the disk and thus provide a good signal-to-noise ratio. In magneto-optic drives, the read signal reflected off the disk is not intensity modulated but *polarization* modulated. Thus polarization optics and the technique of differential detection must be used to convert the *polarization* modulation to intensity modulation (as is discussed later in the section on magneto-optic recording).

© 2006 by Taylor & Francis Group, LLC

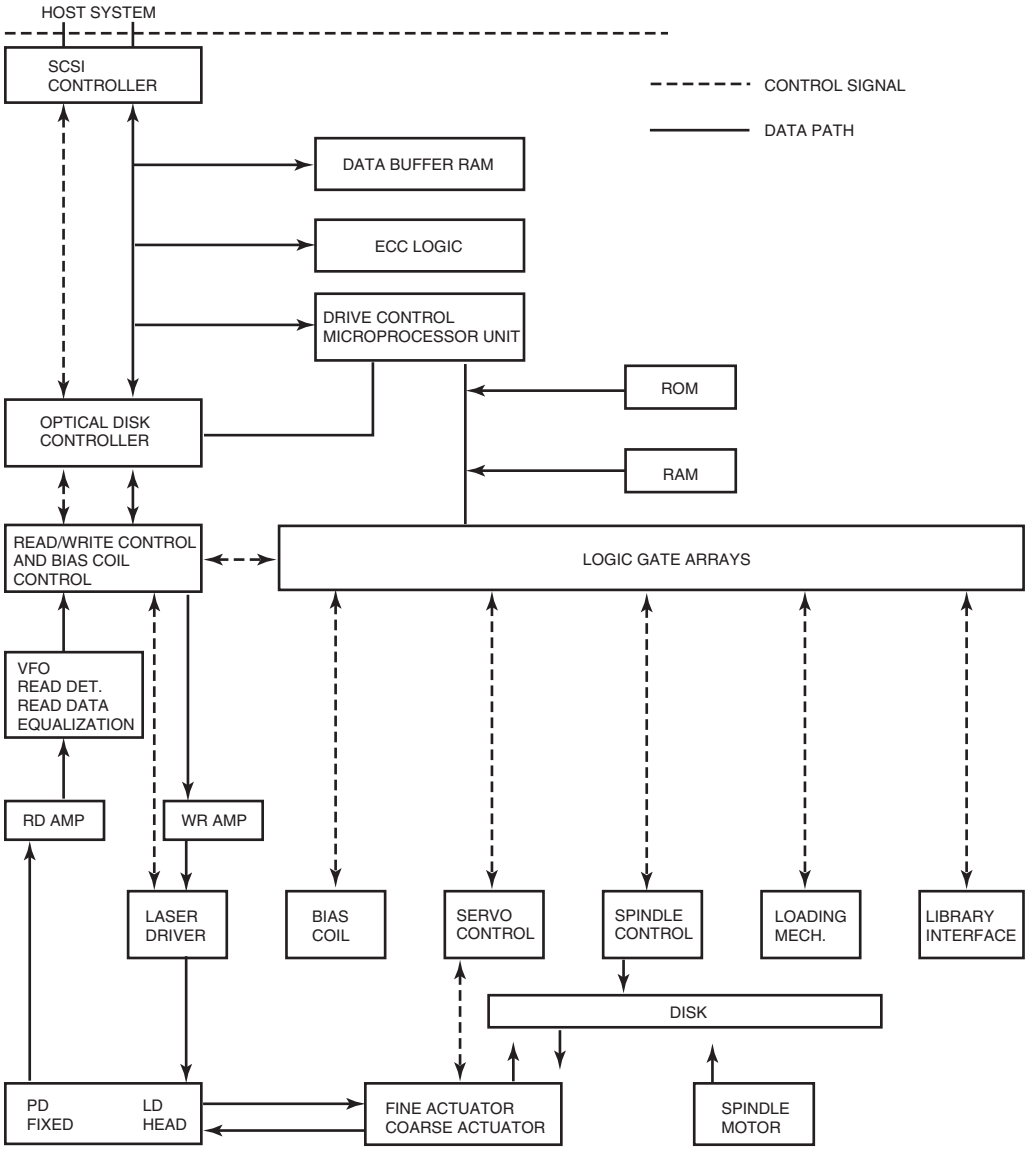


FIGURE 25.4 A functional block diagram of an optical disk drive describing the key electrical functions and interfaces. Some of the abbreviations are: VFO is the variable field oscillator, which is used to synchronize the data, RD amp is the read signal amplifier, WR amp is the write signal amplifier, PD fixed is the photodetextor assemply, and LD head is the laser diode.

To extract the 1s and 0s from the noisy analog signal derived from the photodetectors, optical drives use a number of techniques, such as equalization, which boosts the high frequencies and thus provides greater discrimination between spots. Using an analog-to-digital converter, the analog data signal is converted into channel bits. The channel bits are converted back into customer data bytes using basically the reverse of the encoding process. The data is clocked into the decoder, which removes the modulation code. The remaining special characters are removed from the data, which is then fed into the ECC alignment buffer to correct any errors (up to 40 bytes long). Once data has been read from the disk, it is stored in a RAM buffer and then output to whatever readout device is hooked by SCSI to the drive. With this basic understanding

© 2006 by Taylor & Francis Group, LLC

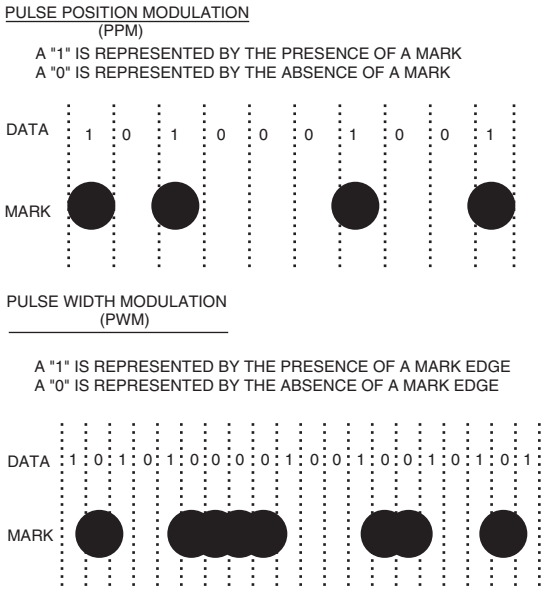

PULSE POSITION MODULATION
(PPM)
A "1" IS REPRESENTED BY THE PRESENCE OF A MARK
A "0" IS REPRESENTED BY THE ABSENCE OF A MARK
DATA
1 0 1 0 0 0 1 0 0 1
MARK
PULSE WIDTH MODULATION
(PWM)
A "1" IS REPRESENTED BY THE PRESENCE OF A MARK EDGE
A "0" IS REPRESENTED BY THE ABSENCE OF A MARK EDGE
DATA
1 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1
MARK

FIGURE 25.5 Schematic showing the mark spacing for PPM and PWM recording. The PPM recording is used in most current optical drives. The PWM recording increases capacity by as much as 50% and will be used in almost all forthcoming writable optical drives. PWM recording, however, requires much tighter tolerances than PPM recording.

of how data is recorded on a spinning disk, we can turn to the specific items such as recording physics that delineate the various recording technologies.

Phase-Change Recording

Phase-change recording takes advantage of the fact that certain materials can exist in multiple *metastable* (i.e. normally stable) crystalline phases, each of which have differing optical properties (such as reflectivity). Thermal energy (as supplied by the focused beam of a high-power laser) above some threshold can be used to switch from one metastable state to another (Ovshinsky, 1970; Takenaga et al., 1983). Energy below the switching threshold should have no effect. In this way a low-power focused spot can be used to read out the recorded information without affecting it.

To achieve this kind of multiple metastable states, phase-change materials typically are a mixture of several elements such as germanium, tellurium, and antimony ($Ge_2Sb_2Te_5$). Phase-change materials are available that are suited for either rewritable or write-once recordings. In an erasable material, recording is affected by melting the material under the focused spot and then cooling it quickly enough to freeze it in an amorphous phase. Rapid cooling is critical and, thus, the design of the heat sinking capability of the material is important.

Erasure of the phase-change material is achieved by an annealing process, that is, heating the material to just below the melting point for a long enough period to recrystallise the material and erase any amorphous marks.

The fact that phase-change materials are a mixture of several materials makes recyclability difficult to achieve. The melting/annealing processes increase phase segregation and thus reduce the number of cycles that can be achieved. Early phase-change materials could only achieve a few thousand cycles, which is one of the primary reasons they were passed over by many companies in favor of magneto-optical products when rewritable drives were being introduced. However, the cyclability of phase-change materials has increased substantially.

The advantage of phase-change recording over magneto-optic recording is that a simpler head design is possible (since no bias magnet and fewer polarizing optics are needed). The disadvantages include the fact

© 2006 by Taylor & Francis Group, LLC

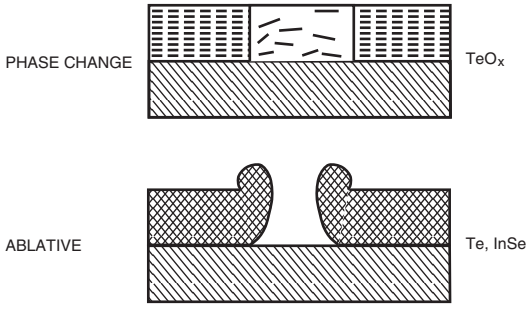


FIGURE 25.6 Permanent WORM recording provides the highest level of data security available in a removable storage device. In ablative WORM recording, marks are physically burned into the material. In phase-change worm, the recording process is a physical change in the material, which results in a change in reflectivity.

that there is less standardization support for the phase-change format, and fewer companies produce the drives. For the consumer this means there is less interchange with phase-change than with magneto-optic.

25.3 Worm Technology

The earliest writable optical drives were, in fact, WORM drives, and although the rewritable drive is more popular for daily storage needs, the WORM technology has a clear place in data storage because it allows permanent archiving capability.

There are a number of different types of write-once technologies that are found in commercial products. Ablative WORM disks consist of tellurium-based alloys. Writing of data is accomplished by using a high-powered laser to burn a hole in the material (Kivits et al., 1982). A second type of WORM material is what is known as textured material, such as in a moth's eye pattern. The actual material is usually a platinum film. Writing is accomplished by melting the textured film to a smooth film and thus changing the reflectivity. Phase-change technology provides a third type of WORM technology using materials such as tellurium oxide. In the writing process, amorphous (dark) material is converted to crystalline (light) material by application of the heat from a focused laser beam (Wrobel et al., 1982). The change cannot be reversed. A comparison of phase-change and ablative WORM recording is shown schematically in Fig. 25.6.

25.4 Magneto-Optic Technology

In a magneto-optic drive, data recording is achieved through a thermomagnetic process (Mayer, 1958), also known as *Curie point writing* as it relies on the threshold properties of the Curie temperature of magnetic materials. In this process, the energy within the focused optical spot heats the recording material past its Curie point (about 200°C), a threshold above which the magnetic domains of the material are susceptible to moderate (about 300 G) external magnetic fields. Application of an external magnetic field is used to set the state of the magnetization vector (which represents the polarization of the magnetic domains) in the heated region to either up (a one bit) or down (a zero bit). When the material is cooled to below the Curie point this orientation of the magnetic domains is fixed. An illustration of the magneto-optic recording process is given in Fig. 25.7. This recording cycle has been shown to be highly repeatable (>1 million cycles) in any given region without degradation of the material. This is an important aspect if the material is to be claimed as fully rewritable.

In any practical recording process, it is necessary to have a sharp threshold for recording. This ensures the stability of the recorded information both to environmental conditions as well as during readout. Thermo-magnetic recording is an extremely stable process. Unless heated to high temperatures (>100°C), the magnetic domains in magneto-optic recording films are not affected by fields under several kilogauss

© 2006 by Taylor & Francis Group, LLC

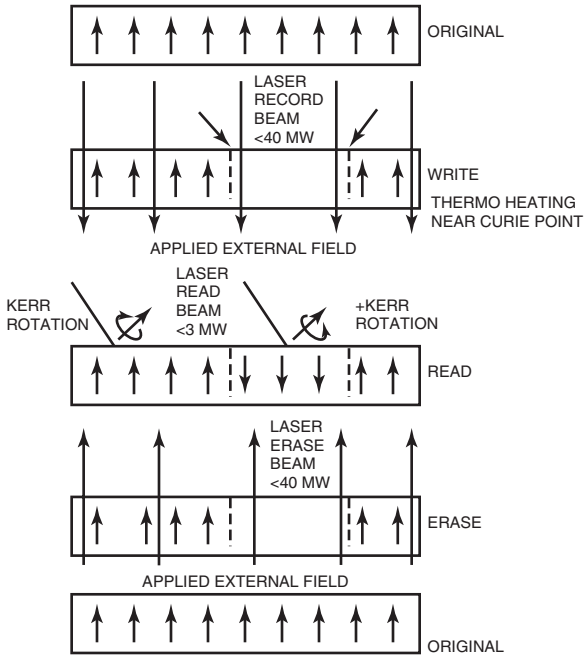


FIGURE 25.7 Schematic showing the recording, readout, and erasure processes of a magneto-optic material.

in strength (in comparison, the information stored on a magnetic floppy is affected by magnetic fields as low as a 100 G). The coercivity of a magneto-optic material remains high until very close to the Curie temperature. Near the Curie temperature (about 200°C), the coercivity rapidly drops by two or three orders of magnitude as the magnetic domain structure becomes disordered.

Readout of the recorded information can safely be achieved with a laser beam of about 2-mW power at the disk, a power level which is high enough to provide good signal strength at the detectors, but low enough not to affect the recorded information because any media heating from it is far below the Curie threshold. During readout, the magnetic state of the recorded bits of information is sensed through the *polar-Kerr effect* by a low-power linearly polarized readout beam. In this effect the plane of polarization of the light beam is rotated slightly (0.5°) by the magnetic vector. The direction of rotation, which defines whether the bit is a one or a zero, is detected by the readout detectors and channel. Although the tiny amount of Kerr rotation results in a very small amount of signal modulation riding on a large DC bias, the technique of differential detection permits acceptable signal-to-noise ratio (SNR) to be achieved.

The output signal in an MO recording system is the signal from the light falling on one detector minus the signal from the light falling on the other detector. By placing a polarizing beamsplitter at 45° to the incident polarization, the two data detectors get the signals (Mansuripur, 1982)

$$\begin{aligned} d_1 &= \frac{I_0}{2}(\cos\theta_k/2) - \sin(\theta_k/2))^2 \approx \frac{I_0}{2}(1-\theta_k) \\ d_2 &= \frac{I_0}{2}(\cos\theta_k/2) + \sin(\theta_k/2))^2 \approx \frac{I_0}{2}(1+\theta_k) \end{aligned} \tag{25.2}$$

in which d_1 and d_2 refer to the detector signals, I_0 is the incident intensity, and $\theta_k/2$, the rotation angle, is assumed to be small. The readout signal is taken as

$$s = (d_1 - d_2)/(d_1 + d_2) \tag{25.3}$$

As can be seen, this signal does not contain intensity noise from either the laser or from reflectivity variations of the disk. But the signal is very sensitive to polarization noise, which may be introduced by polarization

© 2006 by Taylor & Francis Group, LLC

sensitive diffraction, by substrate birefringence effects, by inhomogeneities in the magnetooptic films, or by other polarization sensitive components.

Early MO recording media, such as manganese bismuth (MnBi) thin films (Chen et al., 1968), were generally crystalline in nature. The magnetic domains followed the crystalline boundaries and thus were irregular in shape (Marchant, 1990). The crystalline nature of the films caused optical scattering of the readout signal, and the irregular domains led to noise in the recorded signal. The combination degraded the SNR sufficiently to make polycrystalline magneto-optic media impractical.

The discovery in 1976 of magneto-optic materials based on the rare earth/transition metal (RE/TM) alloys (Choudhari et al., 1976) provided a practical material system for rewritable magneto-optic recording. These materials were amorphous and thus allowed acceptable signal-to-noise ratio to be obtained. Most commercial magneto-optic films today are based on terbium iron cobalt (TbFeCO).

25.5 Compact Disk-Recordable (CD-R)

The writable version of the popular CD-R disk looks very much like a stamped CD-ROM disk, and can be played in most CD-ROM players.

One of the reasons why CD-audio and CD-ROM have become so successful is the strict, uniform standards that all of the manufacturers of these products have adhered to. The standards were drawn up by Philips and Sony and are described by the colors of the books that they were first printed in. The standards describing CD-audio are found in the Red Book, those describing CD-ROM are found in the Yellow book, and those describing CD-R are found in the Orange Book. These books describe the physical attributes that the disks must meet (such as reflectivity, track pitch, etc.), as well as they layout of recorded data (Bouwhuis et al., 1985).

In prerecorded CD-ROM disks, the information is stamped as low reflectivity pits on a high reflectivity background. The disk has a reflectivity of 70% (achieved by using an aluminum layer), whereas the pits have a reflectivity of 30% (these reflectivity specifications have been defined in the Red Book). The CD-ROM drive uses this difference in reflectivity to sense the information stamped on the disk. To be compatible with CD-ROM readers, a CD-R disk must also use this reflectivity difference when recording data. To accomplish this, a CD-R disk is coated with an organic polymer that can change its local reflectivity permanently upon sufficient heating by a laser spot. The structure of a CD-R disk is shown in Fig. 25.8.

When the organic dye polymer is locally heated by the focused spot of a laser beam, polymeric bonds are broken or altered resulting in a change in the complex refractive index within the region. This refractive index change results in a change in the material reflectivity. There are a half-dozen organic dye polymers that are commercially being used. Two examples are phthalocyanine and poly-methane cyanine.

Like the CD-ROM drives, the CD-R drives have relatively low performance (when compared with optical or hard drives). The seek times are on the order of a few hundred milliseconds, whereas the maximum

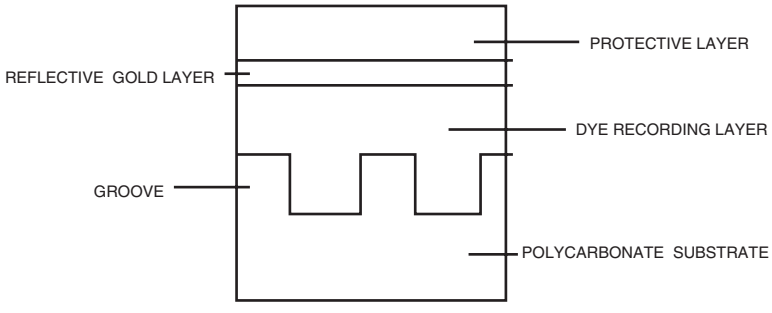


FIGURE 25.8 The basic structure of a CD-R disk.

© 2006 by Taylor & Francis Group, LLC

data rate for a 4X speed drive is about 600 kilobytes/s. The seek time is slow because the CD-R drives spin the disks in *constant linear velocity* (CLV) mode as defined in the Red Book standards. Constant linear velocity means that the disk rotation speed varies with the radius at which the read head is positioned in such a way as to ensure that the linear velocity is constant with radius.

Pure data devices such as hard disks and optical WORM drives, however, can handle constant angular velocity (CAV) operation in which the linear velocity and data rate increase with radial position. The disadvantage of CLV operation is that it results in a very slow access time. When a seek of the optical head is performed, the motor speed has to be adjusted according to the radial position of the head. This takes time and thus lengthens the seek operation to 250–300 ms. In contrast, constant angular velocity devices like optical WORM disks have seek times on the order of 40 ms.

The roots of CD-R are found in audio CD, and thus some of the parameters, recording formats, and performance features are based on the Red Book standard (which has tended to be a handicap). The CD format (Bouwhuis et al., 1985) is not well suited for random access block oriented recording. The CD format leads to a sequential recording system, much like a tape recorder.

Recording Modes of CD-R

To understand the attributes and limitations of CD-R, it is important to understand the various recording modes that it can operate in. For fixed block architecture devices, the question of recording modes never comes up as there is only one mode, but in CD-R, there are four modes (Erlanger, 1994).

The four recording methods in CD-R drives are

- Disk-at-once (or single session)
- track-at-once
- Multisession
- Incremental packet recording

In disk-at-once recording, one recording session is allowed on the disk, whether it fills up the whole disk or just a fraction of the disk. The data area in a single session disk consists of a lead-in track, the data field, and a lead out track. The lead-in track contains information such as the table of contents (TOC). The lead-in and lead-out are particularly important for interchange with CD-ROM drives. In single session writing, once the lead-in and lead-out areas are written, the disk is considered finalized and further recording (even if there are blank areas on the disk) cannot take place. After the disk is finalized, it can be played back on a CD-ROM player (which needs the lead-in and lead-out tracks present just to read the disk).

Having just the capability of recording a single session can be a quite a limitation for obvious reasons, and so the concept of *multisession* recording was introduced. An early proponent of multisession recording was Kodak, which wanted multisession capability for its photo-CD products. In multisession recording, each session is recorded with its own lead-in and lead-out areas. Multisession recorded disks can be played back in CD-ROM drives that are marked multisession compatible (assuming that each session on the disk has been finalized with lead-in and lead-out areas). Unfortunately, the lead-in and lead-out areas for each session take up lots of overhead (about 15 megabytes). With this kind of overhead, the ultimate maximum number of sessions that can be recorded on a 650 megabyte disk is 45 sessions.

Rather than do multisession recording, the user may choose track-at-once recording. In this type of recording, a number of tracks (which could represent distinct instances of writing) could be written within each session. The maximum number of tracks that can be written on the whole disk is 99. However, the disk or session must be finalized before it can be read on a CD-ROM drive.

Because of the way input data are encoded and spread out, it is imperative to maintain a constant stream of information when recording. If there is an interruption in the data stream, it affects the whole file being recorded (not just a sector as in MO or WORM drives). If the interruption is long enough, it will cause a blank region on the disk and will usually lead to the disk being rendered useless.

Many of these problems or inconveniences can be alleviated through a recording method called *packet recording*. In packet recording, the input data are broken up into packets of specified size (for example 128

© 2006 by Taylor & Francis Group, LLC

kilo-bytes or 1 megabyte). Each packet consists of a link block, four run-in blocks, the data area, and two run-out blocks. The run-in and run-out blocks help delineate packets and allow some room for stitching, that is, provide some space for overlap if perfect synching is not achieved when recording an adjacent packet in a different CD-R drive.

Packet recording has several advantages. To begin with, there is no limit to the number of packets that can be recorded (up to the space available on the disk, of course), and so limitations imposed by track-at-once, multisession, or disk-at-once can be avoided. Also, if the packet size is smaller than the drive buffer size (as is likely to be the case), a dedicated hard drive is not needed while recording. Once the packet of information has been transferred to the drive buffer, the computer can disengage and do other tasks while the CD-R drive performs the recording operation.

With the advent of packet recording, CD-R technology becomes much more flexible than in the past and thus more attractive as a general purpose removable data storage device. It can be used for backup purposes as well the storage of smaller files. However, there is a problem of interchange with CD-ROM players. Some CD-ROM players cannot read a CD-R disk that has been packet written because they post a hard error when they encounter the link block at the beginning of each packet. Packet written CD-R disks can be read on CD-R drives and on CD-ROM drives that are packet enabled.

25.6 Optical Disk Systems

Disks

An important component of the optical storage system is, of course, the media cartridges. In fact, the greatest attraction of optical storage is that the storage media is removable and easily transported, much like a floppy disk. Most of the writable media that is available comes in cartridges of either 3.5- or 5.25-in. form factors. An example of the 5.25-in. cartridge is shown in Fig. 25.9. The cartridge has some sensory holes to allow its media type to be easily recognized by the drive. The tracks on the media are arranged in a spiral fashion. The first block of data is recorded on the innermost track (i.e., near the center). Recordable CD media is cartridgeless and is played in CD-R/CD-ROM drives either using a caddy or through tray loading (like audio CD players).

Cartridged media has been designed to have a long shelf and archival life (without special and expensive environmental control) and to be robust enough to survive the rigors of robotic jukeboxes and customer handling. Magneto-optic and WORM media is extremely stable and so data can be left on the media with great confidence (Okino, 1987). A chart showing projected lifetimes (after accelerated aging) of IBM WORM media is given in Fig. 25.10, which indicates that the lifetimes for 97.5% of the media surfaces for shelf and archival use are projected to exceed 36 and 510 years, respectively, for storage at 30°C/80%

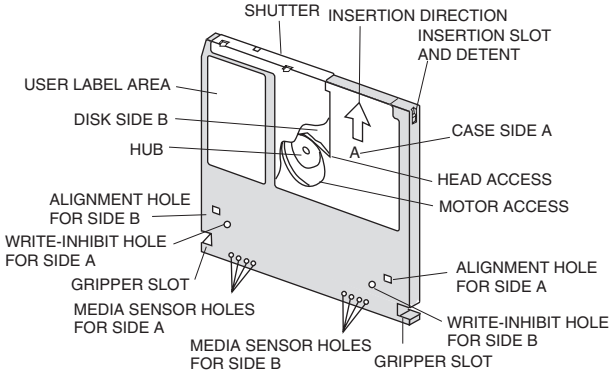


FIGURE 25.9 Schematic showing a 5.25-in disk cartridge. The various slots in the cartridge can be sensed by the drive and thus provide information about the disk to the drive.

© 2006 by Taylor & Francis Group, LLC

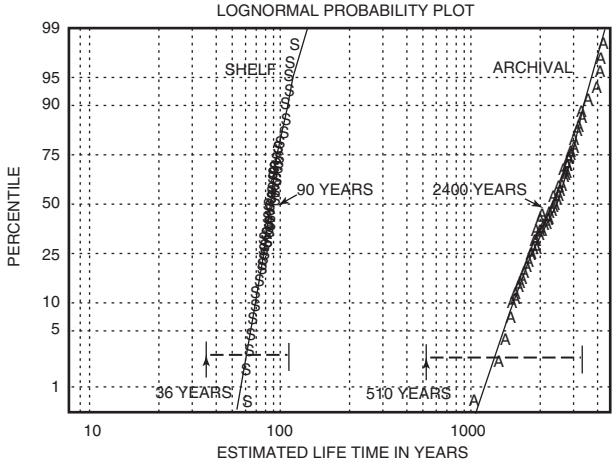


FIGURE 25.10 Estimated lifetime of an IBM WORM disk. Lognormal plot of lifetimes for shelf and archival use. (*Source:* Wong, J.S. et al., 1993. Life expectancy of IBM write-once media. IBM White Paper, IBM Publications.)

relative humidity. Shelf life is the length of time that data can be effectively written, whereas archival life is the length of time that data can be effectively read.

Optical media is perhaps the most stable digital storage technology: magnetic drives are prone to head crashes, tape deteriorates or suffers from print through (in which information on one layer is transferred to another layer in the tightly wound tape coil), and paper or microfiche also deteriorate with time (Rothenberg, 1995).

Automated Optical Storage Systems

Optical drives can be extended to automated storage systems, which are essentially jukeboxes consisting of one or more optical drives and a large number of optical disk cartridges. Optical libraries can provide on-line, direct-access, high-capacity storage.

Applications of Optical Library Systems

Optical library systems fit well into a large computer-based environment, such as client-server systems, peer-to-peer local area networks, or mainframe-based systems. In such environments, there is a distinct storage hierarchy based on cost/access trade-off of various types of data. This hierarchy is shown schematically in Fig. 25.11 as a pyramid. The highest section of the pyramid contains the highest performance and highest cost type of memory. The most inexpensive (on a cost/megabyte) basis and lowest performance is that of tape. Optical libraries are an important part of this segment because they provide storage with performance capabilities approaching that of magnetic, but at a cost approaching that of tape.

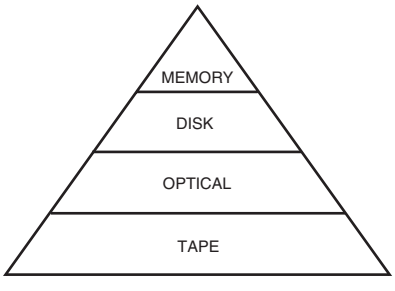


FIGURE 25.11 The storage hierarchy.

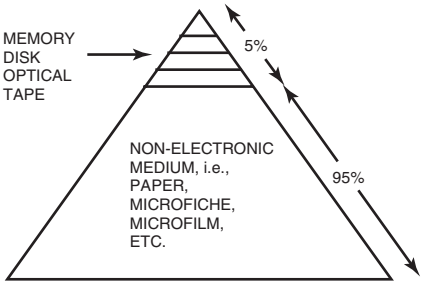
An optical library contains a cartridge transport mechanism called an autochanger. The autochanger moves optical cartridges between an input/output slot (through which cartridges can be inserted into the library), the drives (where the cartridges are read or written), and the cartridge storage cells.

Sophisticated storage systems provide a hierarchical storage management capability in which data are automatically migrated between the various layers of the pyramid depending on the access needs of the

© 2006 by Taylor & Francis Group, LLC

data. For example, a telephone company can keep current billing information on magnetic storage, whereas billing information older than a month can be stored in optical libraries, and information older than six months can be stored on tape. It does not make sense to keep data that the system does not frequently access on expensive storage systems such as semiconductor memory or high-performance magnetic drives.

One of the applications for which automated optical storage is ideally suited is document imaging. Document imaging directly addresses the substantial paper management problem in society today. The way we store and manage paper documents has not changed significantly in over a century, and largely does not take advantage of the widespread availability of computers in the workplace. Trying to retrieve old paper documents is one of the most inefficient of business activities today (some businesses have stockrooms full of filing cabinets or document boxes). The way information is currently stored can be represented by a pyramid (as shown in Fig. 25.12), in which paper accounts for almost all of the document storage.



SOURCE: ASSOCIATION FOR IMAGE AND INFORMATION MANAGEMENT (AIIM), 1990

FIGURE 25.12 The paper pyramid. Most of the world's documents are in the form of paper, only 5% are in electronic form.

An application aimed squarely at re-engineering the way we handle and store documents is document imaging. In document imaging, documents are scanned onto the computer and stored as computer readable documents. Storing documents in computer format has many advantages, including rapid access to the information. Recall of the stored documents is as easy as typing in a few key search words. Document imaging, however, is very storage intensive and, thus, a high-capacity, low-cost storage technology is required, ideally with fast random access. Optical storage is now recognized as the optimal storage technology in document imaging. Optical libraries can accommodate the huge capacities required to store large numbers of document images (text and/or photos), and optical drives can provide random access to any of the stored images. Figure 25.13 illustrates the equivalent storage capacity of a single 1.3 gigabyte optical disk.

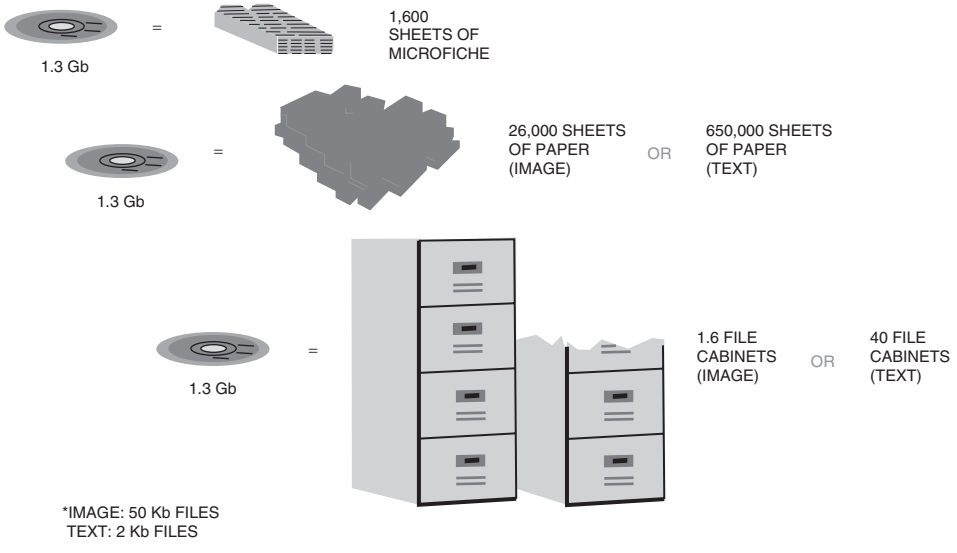


FIGURE 25.13 Equivalent storage capacity of a single 1.3 gigabyte optical disk and microfiche, paper, and filing cabinets.

© 2006 by Taylor & Francis Group, LLC

Future Technology and Improvements in Optical Storage

Optical drives and libraries, like anyhigh-technology products, will see two types of improvement processes. The first is an incremental improvement process in which quality and functionality will continuously improve. The second is a more dramatic improvement process in which disk capacity and drive performance will improve in distinct steps every few years.

For the library, the improvements will be in the speed of the automated picker mechanism as well as the implementation of more sophisticated algorithms for data and cartridge management. Most of the improvements in the library systems will arise out of improvements in the drives themselves.

For the optical drive, the incremental improvement process will concentrate on the four core technology elements: the laser, the media, the recording channel, and the optomechanics. The laser will see continual improvements in its beam quality (reduction of wavefront aberrations and astigmatism), lifetime (which will allow drive makers to offer longer warranty periods), and power (which will allow disks to spin faster). The media will see improvements in substrates (reduction of tilt and birefringence), active layers (improved sensitivity), and passivation (increased lifetimes). The optics and actuator system will see improvements in the servosystem to allow finer positioning, reductions in noise, smaller optical components, and lighter actuators for faster seek operations. The recording channel and electronics will see an increase in the level of electronics integration, a reduction in electronic noise, the use of lower power electronics, and better signal processing and ECC to improve data reliability. One of the paramount directions for improvement will be in the continuous reduction of cost of the drives and media. This step is necessary to increase the penetration of optical drives in the marketplace.

In terms of radical improvements, there is a considerable amount of technical growth that is possible for optical drives. The two primary directions for future work on optical drives are (1) increasing capacity and (2) improving performance specifications (such as data rate and seek time). The techniques to achieve these are shown schematically in Fig. 25.14.

The performance improvements will include spinning the disk at much higher revolutions per minute to get higher data rates, radically improving the seek times for faster access to data, and implementing direct overwrite and immediate verify techniques to reduce the number of passes required when recording. Finally, the use of parallel recording (simultaneous recording or readback of more than one track), either through the use of multielement lasers or through special optics, will improve the data rates significantly.

In optical disk products, higher capacities can only be achieved by higher areal densities since the size of the disk cannot increase from presently established standard sizes. There are a number of techniques that are being considered for improving the storage capacity of optical drives. These include

- Use of shorter wavelength lasers or *superresolution* techniques to achieve smaller spot sizes
- Putting the data tracks closer together for greater areal density

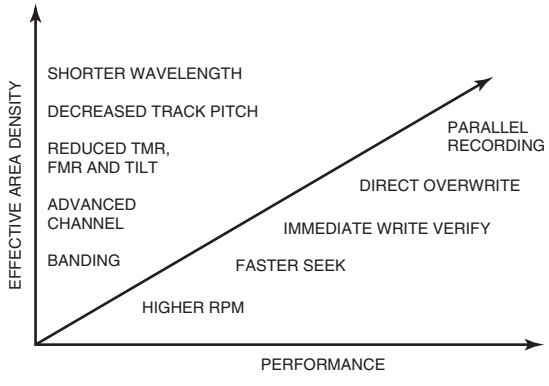


FIGURE 25.14 Technological directions to achieve performance and capacity improvements in optical drives.

© 2006 by Taylor & Francis Group, LLC

- Reducing the sensitivity to *focus misregistration* (FMR) (i.e., misfocus) and *tracking misregistration* (TMR) (i.e., when the focused spot is not centered on the track)
- Reducing the sensitivity to media tilt

Finally, improvements in the read channel such as the use of partial-response-maximum-likelihood (PRML) will enable marks in high-density recording to be detected in the presence of noise.

Acknowledgments

The author is grateful to two IBM Tucson colleagues: Blair Finkelstein for discussions and suggestions on the read channel section and Alan Fennema for writing some of the paragraphs in the servo section.

Defining Terms

Compact disk erasable/compact disk recordable (CD-E/CD-R): These define the writable versions of the compact disk. CD-E is a rewritable media and CD-R is a write-once media.

Data block size: The data that is to be recorded on an optical disk is formatted into minimum block sizes. The standard block size, which defines a single sector of information, is 512 byte. Many DOS/windows programs expect this block size. If the block size can be made larger, storage usage becomes more efficient. The next jump in block size is 1024 byte, often used in Unix applications.

Device driver: This is a piece of software that enables the host computer to talk to the optical drive. Without this piece of software, you cannot attach an optical drive to a PC and expect it to work. As optical drives grow in popularity, the device driver will be incorporated in the operating system itself. For example, the device driver for CD-ROM drives is already embodied in current operating systems.

Error correction and control (ECC): These are codes (patterns of data bits) that are added to raw data bits to enable detection and correction of errors. There are many types of error correcting codes. Compact disk drives, for example, use cross interleaved Reed–Solomon codes.

Magneto-optical media: An optical recording material in which marks are recorded using a thermomagnetic process. That is, the material is heated until the magnetic domains can be changed by the application of a modes magnetic field. This material is rewritable.

Optical jukebox: This is very similar to a traditional jukebox in concept. A large number of disks are contained in a jukebox and can be accessed at random for reading or writing.

Phase-change media: An optical recording material consisting of an alloy that has two metastable phases with different optical properties. Phase-change media can be rewritable or write-once.

Pulse-position modulation (PPM): A recording technique in which a mark on the disk signifies a binary 1 and its absence signifies a binary 0. A 1001 bit sequence is mark-space-space-mark.

Pulse-width modulation (PWM): A recording technique in which the edges of the mark represent the ones and the length of the mark represents the number of zeros. Thus, a 1001 sequence is represented by one mark.

Seek time/access time: The two terms are often used interchangeably, which is incorrect. The seek time is, by convention, defined as the length of time taken to seek across one-third of the full stroke of the actuator (which is from the inner data band to the outer data band on the disk). The access time is the seek time plus some latency for settling of the actuator and for the disk to spin around appropriately. The access time really states how quickly you can get to data.

Servosystem: The mechanism, which through feedback and control, keeps the laser beam on track and in focus on the disk—no easy task on a disk spinning at 4000 rpm.

Tracking: The means by which the optical stylus (focused laser beam) is kept in the center of the data tracks on an optical disk.

References

Arimoto, A. et al. 1986. Optimum conditions for high frequency noise reduction method in optical videodisk players. *Appl. Opt.* 25(9):1.

Asthana, P. 1994. A long road to overnight success. *IEEE Spectrum* 31(10):60.

© 2006 by Taylor & Francis Group, LLC

Bouwhuis, G., Braat, J., Huijser, A., Pasman, J., van Rosmalem, G., and Schouhamer Immink, K. 1985. *Principles of Optical Disk Systems.* Adam Hilger, Bristol, England, UK.

Braat, J. and Bouwhuis, G. 1978. Position sensing in video disk read-out. *Appl. Opt.* 17:2013.

Chen, D., Ready, J., and Bernal, G. 1968. MnBi thin films: Physical properties and memory applications. *J. Appl. Phys.* 39:3916.

Choudhari, P., Cuomo, J., Gambino, R., and McGuire, T. 1976. U.S. Patent #3,949,387.

Earman, A. 1982. Optical focus servo for optical disk mass data storage system application. *SPIE Proceedings* 329:89.

Erlanger, L. 1994. Roll your own CD. *PC Mag.* (May 17):155.

Golomb, S. 1986. Optical disk error correction. *BYTE* (May):203.

Goodman, J. 1968. *Introduction to Fourier Optics*. McGraw-Hill, San Francisco.

Inoue, A. and Muramatsu, E. 1994. Wavelength dependency of CD-R. *Proceedings of the Optical Data Storage Conference.* Optical Society of America, May, p. 6.

Kivits, P., de Bont, R., Jacobs, B., and Zalm, P. 1982. The hole formation process in tellerium layers for optical data storage. *Thin Solid Films* 87:215.

Mansuripur, M., Connell, G., and Goodman, J.W. 1982. Signal and noise in magneto-optical readout. *J. Appl. Phys.* 53:4485.

Mansuripur, M. 1987. Analysis of astigmatic focusing and push-pull tracking error signals in magneto-optical disk systems. *Appl. Opt.* 26:3981.

Marchant, A. 1990. *Optical Recording*. Addison-Wesley, Reading, MA.

Mayer, L. 1958. Curie point writing on magnetic films. *J. Appl. Phys.* 29:1003.

Okino, Y. 1987. Reliability test of write-once optical disk. *Japanese J. Appl. Phys.* 26.

Ovshinsky, S. 1970. Method and apparatus for storing and retrieving information. U.S. Patent #3,530,441.

Rothenberg, J. 1995. Ensuring the longevity of digital documents. *Sci. Am.* (Jan.).

Sukeda, H., Ojima, M., Takahashi, M., and Maeda, T. 1987. High density magneto-optic disk using highly controlled pit-edge recording. *Japanese J. Appl. Phys.* 26:243.

Takenaga, M. et al. 1983. New optical erasable medium using tellerium suboxide thin film. *SPIE Proc.* 420:173.

Tarzaiski, R. 1983. Selection of 3f (1,7) code for improving packaging density on optical disk recorders. *SPIE Proc.* 421:113.

Treves, D. and Bloomberg, D. 1986. Signal, noise, and codes in optical memories. *Optical Eng.* 25:881.

Wrobel, J., Marchant, A., and Howe, D. 1982. Laser marking of thin organic films. *Appl. Phys. Lett.* 40:928.

Further Information

There are a number of excellent books that provide an overview of optical disk systems. A classic is *Optical Recording* by Alan Marchant (Addison-Wesley, Reading, MA, 1990), which provides an overview of the various types of recording as well as the basic functioning of an optical drive. A more detailed study of optical disk drives and their opto-mechanical aspects is provided in *Principles of Optical Disc Systems* by G. Bouwhuis, J. Braat, A. Huijser, J. Pasman, G. van Rosmalen, and K. Schouhamer Immink (Adam Hilger Ltd., Bristol, England, 1985). An extensive study of magneto-optical recording is presented in *The Physical Properties of Magneto-Optical Recording* by Masud Mansuripur (Cambridge University Press, London, 1994).

For recent developments in the field of optical storage, the reader is advised to attend the meetings of the International Symposium on Optical Memory (ISOM) or the Optical Data Storage (ODS) Conferences (held under the auspices of the IEEE or the Optical Society of America).

For information on optical storage systems and their applications, a good trade journal is the *Computer Technology Review*. *Imaging Magazine* usually has a number of good articles on the applications of optical libraries to document imaging, as well as periodic reviews of commercial optical products.

© 2006 by Taylor & Francis Group, LLC

26

Error Correction

26.1 Background **26**-1
26.2 Introduction **26**-2
26.3 The Development of Error Correction Codes **26**-3
26.4 Code Families **26**-3
26.5 Linear Block Codes **26**-4
26.6 Convolutional Codes **26**-6
26.7 Trellis Coded Modulation **26**-9
26.8 Applications **26**-9

Fabrizio Pollara

26.1 Background

Digital data transmitted over channels with impairments such as noise, distortions, and fading is inevitably delivered to the user with some errors. A similar situation occurs when digital data is stored on devices such as magnetic or other media that contain imperfections. The rate at which errors occur is a very important design criterion for digital communication links and for data storage. Usually, it must be kept smaller than a given preassigned value, which depends on the application. Error correction techniques based on the addition of **redundancy** to the original message can be used to control the error rate. The most obvious, albeit not efficient, example is the repetition of the same message. In this chapter we illustrate several efficient error correction techniques.

In 1948 Claude Shannon revolutionized communication technology with the publication of his classic paper in which he showed that the most efficient solution for reliable communications necessarily involved *coding* (i.e., introducing a controlled amount of redundancy) the selected message, before transmission over the channel. Shannon did not say exactly how this coding should be done: he only proved mathematically that efficient coding schemes must exist. Since 1948, a whole generation of later researchers has validated Shannon's work by devising explicit and practical coding schemes, which are now part of nearly every modern digital communication system. Satellite communication systems, high-performance military systems, computer communication networks, high-speed modems, and compact-disk and magnetic recording and playback systems, all rely heavily on sophisticated coding schemes to enhance their performance.

Shannon showed that to every communication channel we can associate a **channel capacity** C, and that there exist error correcting codes such that information can be transmitted across a noisy channel at rates less than C, with arbitrarily low bit error rate. In fact, an important implication of Shannon's theory is that it is wasteful to build a channel that is too good; it is more economical to use an appropriate code. Until Shannon's discovery, it was believed that the only way to overcome channel noise was to use more powerful transmitters or build larger antennas.

A communication system connects a data source to a data user through a channel. Microwave links, coaxial cables, telephone circuits, and even magnetic tapes are examples of channels. Coded digital communication systems (or data storage systems) include a device that creates appropriate redundant bits (the encoder) and a corresponding device (the decoder) that uses this redundancy to correct some of the

© 2006 by Taylor & Francis Group, LLC

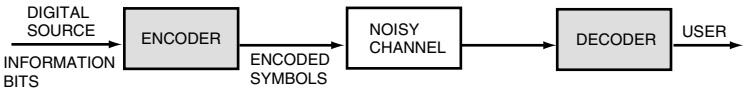


FIGURE 26.1 Coded communication system.

errors introduced by the channel, as shown in Fig. 26.1. The encoder takes a block of information bits and produces a larger block of channel symbols, which is called a **codeword**. The channel in Fig. 26.1 contains all of the devices necessary to transmit digital information on a physical medium, including: a modulator to convert each symbol of the codeword into a corresponding analog symbol from a finite set of possible analog symbols (waveforms), and a demodulator to convert each received channel output signal into one of the codeword symbols. Each demodulated symbol is a best estimate of the transmitted symbol, but the demodulator makes some errors because of channel noise.

26.2 Introduction

The basic ideas of efficient error correction are illustrated in the following example of a *Hamming code*, a single-error-correcting code belonging to the first class of practical codes invented by Richard Hamming in 1948.

> Example of single-error-correcting code: Hamming code. The data to be transmitted consists of blocks of four bits denoted by $\{d_1, d_2, d_3, d_4\}$. Three redundant bits $\{p_1, p_2, p_3\}$, the parity-check bits, are computed from the bits in the data block and are appended to this block by the *encoder*. The parity-check p_1 is set to the value 0 if the number of 1s among the bits d_2, d_3, and d_4 is even and is set to 1 otherwise. Similarly, p_2 is set to 0 or 1 by the parity, that is, the even- or oddness, of the number of 1s among d_1, d_3, and d_4. Finally, p_3 is determined by the parity of d_1, d_2, and d_4. For example, if the data bits are {1011}, then the parity bits will be {010}. The combination of the data bits and the parity bits (seven bits, in this example) is sent over the channel or stored in a memory device and is called the codeword.
>
> When the codeword is recovered, possibly with some bit errors, the decoder checks to see if the three parity-check rules are still satisfied. The decoders computes a 0 for legitimate parity checks and a 1 for incorrect parity checks, thus generating three bits that are called the **syndrome**. For example, if {1001010} is received, the first syndrome bit is set to 1, because the received data bits $d_2 = 0, d_3 = 0$ and $d_4 = 1$ have odd parity but the first parity-check bit is 0. Similarly, the second and third syndrome bits are set to 1 and 0, to produce the syndrome {110}. This syndrome is then used to find the location of the error using the one-to-one correspondence between syndromes and error locations illustrated in Fig. 26.2, for this example. In our example, the syndrome is {110} which indicates that an error occurred at location d_3, the third data bit. The decoder will then change {1001010} into {1011010} and deliver to the user the corrected data {1011}. This strategy will successfully locate any single error in a block of four data bits encoded with three additional parity bits. If more than one error occurs this method will fail.

More sophisticated coding strategies are available for applications where it is necessary to correct more errors. From a practical standpoint, the essential limitation of all coding and decoding schemes proposed to date has not been a lack of good codes but the complexity (and cost) of the decoder. For this reason, efforts have been directed toward the design of coding and decoding schemes that could be easily implemented.

SYNDROME	000	001	010	011	100	101	110	111
ERROR LOCATION	–	p_3	p_2	d_1	p_1	d_2	d_3	d_4

FIGURE 26.2 Syndrome table.

© 2006 by Taylor & Francis Group, LLC

The Benefits of Error Correction Coding

The amount of redundancy inserted by the code is usually expressed in terms of the **code rate** R. This rate is the ratio of the number of information bits k in a block to the total number of transmitted symbols n in a codeword ($n = k+$ number of redundant symbols, i.e., $n > k$, or, equivalently, $R = k/n < 1$).

Let us assume that the we want to transmit data at a rate of R_b information bits per second. The redundancy of the code forces us to increase the transmitted symbol rate R_s to $R_s = R_b/R > R_b$. If the transmitted energy per information bit E_b is fixed, then the received energy per transmitted symbol is reduced from E_b to $E_s = R \times E_b$. If we did not perform any special decoding operations, the bit error rate (BER) would actually be worse than its original value obtained by using no coding at all! However, if we use the redundancy to correct some of the errors and the code was well selected, the BER at the output of the decoder is better than that at the output of the demodulator of the original uncoded system. This improvement in BER is obtained with the same E_b and R_b of the uncoded system. This explains how coding can improve BER at given E_b and R_b. Alternatively, the effect of coding can be seen as a reduction of required E_b at given BER and R_b, or as an increase in R_b (throughput) at given BER and E_b. The ratio by which the throughput is increased, or the transmitter power can be decreased, is called the *coding gain* of a given coding system.

26.3 The Development of Error Correction Codes

The Hamming codes were disappointingly weak compared with the far stronger codes promised by Shannon's theoretical results. The major advance came when Bose and Ray-Chaudhuri (1960) and Hocquenghem (1959) found a large class of multiple-error-correcting codes (the BCH codes), and Reed and Solomon (1960) found a related class of codes for nonbinary channels. All of these codes were based on algebraic properties of finite fields (**Galois fields**).

A completely different attack to the construction of efficient codes, more probabilistic in flavor, was the discovery in the late 1950s of **convolutional codes**, based on the properties of binary sequences generated by shift registers. These codes gained widespread popularity when an efficient decoding algorithm was discovered in 1967. The use of an algebraic **block code** cascaded with a convolutional code (**concatenated codes**) provided the most powerful codes available until the early 1990s.

The major progress in the 1970s was the discovery of family of codes that are good asymptotically, that is, when the block size of the encoded message becomes arbitrarily large. More efficient and asymptotically good codes, based on algebraic-geometric properties, were discovered in the late 1980s.

Codes that are very efficient for transmission on channels with limited bandwidth were discovered in the early 1980s, by combining the coding operation with an appropriate choice of the modulation scheme, that is, the set of waveforms that are actually used to transmit the encoded symbols. These codes have a major practical significance in the design of virtually any commercial data communication system.

More recently, the development of a new class of codes (**turbo codes**) promises to fill the last gap toward achieving the ultimate limits set by Shannon's theory.

26.4 Code Families

Error correction codes can be divided into two different classes: block codes and convolutional codes. The encoder for a block code breaks the continuous sequence of information bits into k-bit sections or blocks, and then operates on these blocks independently according to the particular code used. With each possible information block is associated an n-tuple of channel symbols, where $n > k$. The quantity n is referred to as the *code length* or *block length*. Codes can be defined on larger than binary alphabets, where each information sample is not just a bit but a symbol taken from an alphabet with q entries.

The other type of codes, called convolutional codes, operate on the information sequence without breaking it up into independent blocks. Rather, the encoder for a convolutional code processes the information continuously and associates each long (perhaps semi-infinite) information sequence with a code sequence containing more symbols.

© 2006 by Taylor & Francis Group, LLC

Block codes and convolutional codes have similar error-correcting capabilities and the same fundamental limitations. In particular, Shannon's fundamental theorems hold for both types of codes.

Which Codes Are Good Codes?

Block codes are judged by three parameters: the block length n, the information length k, and the minimum distance d. The minimum distance is a measure of the amount of difference between the two most similar codewords.

The **Hamming distance** $d(x, y)$ between two q-ary sequences x and y of length n is the number of places in which they differ. The *minimum distance* of a code is the Hamming distance of the pair of codewords with smallest Hamming distance. More refined comparisons among codes make use of the **weight distribution** of a code, which is the list of the Hamming distances of each codeword from a given reference codeword. We will see that for **linear codes** the choice of the reference is irrelevant.

Suppose that a codeword is transmitted, and a single error is made by the channel. Then the received word is at a Hamming distance of 1 from the transmitted codeword. If the distance to every other codeword is larger than 1, then the decoder will properly correct the error if it assumes that the closest codeword to the received word was actually transmitted. In general, if t errors occur and if the distance from the received word to every other codeword is larger than d, then the decoder will properly correct the errors if it assumes that the closest codeword to the received word was actually transmitted. This always occurs if $d \geq 2t + 1$, as illustrated geometrically in Fig. 26.3, where each sphere of radius t consist of all n-tuples within distance t of codeword C, which we think as being the center of the sphere. Nonintersecting spheres of radius t can be drawn about each of the codewords. A received word in a sphere is decoded as the codeword at the center of that sphere. If t or fewer errors occur, then the received word is always in the proper sphere, and the decoding is correct. Some received words that have more than t errors will be in a decoding sphere about another codeword and, hence, will be decoded incorrectly. Other received words that have more than t errors will lie in the interstitial space between decoding spheres. This case can be handled in one of the two following methods: (1) An *incomplete decoder* decodes only those received words lying in a decoding sphere about a codeword. Other received words have more than the allowable number of errors and are declared by the decoder to be unrecognizable (uncorrectable error patterns). (2) A *complete decoder* decodes every received word (whether inside or outside a sphere) by selecting the closest codeword. When more than t errors occur, a complete decoder will often decode incorrectly but occasionally will find the correct codeword. A complete decoder is used when it is preferable to have a best guess of the message than to have no estimate at all.

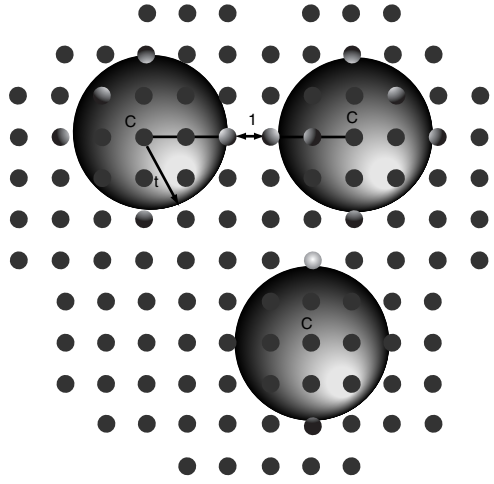


FIGURE 26.3 Geometric representation of a t-error correcting code ($t = 2$).

26.5 Linear Block Codes

Block encoding can be regarded as a table-lookup operation, where each of the $M = q^k$ (q is the alphabet size of information symbols) codewords $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m, \ldots, \boldsymbol{x}_M$ is stored in an n-stage register of a memory bank, and whenever message $\boldsymbol{u}_m = (u_{m1}, \ldots, u_{mk})$ is to be transmitted, the corresponding signal vector $\boldsymbol{x}_m = (x_{m1}, \ldots, x_{mn})$ is read from memory and used as the output of the encoder. We are primarily concerned with binary alphabets, that is, $q = 2$; thus, initially, we take the entries u_{mi} from $\{0, 1\}$ for all m, i.

© 2006 by Taylor & Francis Group, LLC

The linear coding operation for binary data can be represented by

$$\begin{aligned} x_{m1} &= u_{m1}g_{11} \oplus u_{m2}g_{21} \oplus \cdots \oplus u_{mk}g_{k1} \\ x_{m2} &= u_{m1}g_{12} \oplus u_{m2}g_{22} \oplus \cdots \oplus u_{mk}g_{k2} \\ &\vdots \\ x_{mn} &= u_{m1}g_{1n} \oplus u_{m2}g_{2n} \oplus \cdots \oplus u_{mk}g_{kn} \end{aligned} \tag{26.1}$$

where $\oplus$ denotes modulo-2 addition, and $g_{ij} \in \{0, 1\}$ for all i, j. The term $u_{mi}g_{ij}$ is an ordinary multiplication, so that u_{mi} enters into the particular combination for x_{mj} if and only if $g_{in} = 1$. The matrix

$$G = \begin{bmatrix} g_{11}g_{12}\cdots g_{1n} \\ g_{21}g_{22}\cdots g_{2n} \\ \vdots \\ g_{k1}g_{k2}\cdots g_{kn} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix} \tag{26.2}$$

is called the **generator matrix** of the linear code and g_i are its row vectors. Thus Eq. (26.4) can be expressed in vector form as $\boldsymbol{x}_m = \boldsymbol{u}_m G = \sum_{i=1}^{k} u_{mi}g_i$, where $\sum$ means modulo-2 addition and both $\boldsymbol{u}_m$ and $\boldsymbol{x}_m$ are binary row vectors.

An important property of linear codes is that the modulo-2 termwise sum of two codewords $\boldsymbol{x}_m$ and $\boldsymbol{x}_l$ is also a codeword. An interesting consequence of this property is that the set of Hamming distances from a given codeword to the $(M-1)$ other codewords is the same for all codewords.

It can be shown that the generator matrix G can always be rearranged so that the first k entries of each codeword $\boldsymbol{x}_m$ are identical to the k entries of the message $\boldsymbol{u}_m$ and the remaining $n-k$ entries are the parity check symbols, as illustrated in our example of a Hamming code. In this case the code is called **systematic**.

Decoding Linear Block Codes

Suppose the message $\boldsymbol{u} = u_1, \ldots, u_k$ is encoded into the codeword $\boldsymbol{x} = x_1, \ldots, x_n$, which is then sent through the channel. Because of channel noise, the received vector $\boldsymbol{y} = y_1, \ldots, y_n$, may be different from $\boldsymbol{x}$. Let us define the error vector $\boldsymbol{e} = \boldsymbol{y} \oplus \boldsymbol{x} = (e_1, \ldots, e_n)$.

The decoder must decide from $\boldsymbol{y}$ which message $\boldsymbol{u}$ or which codeword $\boldsymbol{x}$ was transmitted. Since $\boldsymbol{x} = \boldsymbol{y} \oplus \boldsymbol{e}$, it is enough if the decoder finds $\boldsymbol{e}$. The decoder can never be certain of what the true $\boldsymbol{e}$ was. Its strategy, therefore, will be to choose the most likely error vector $\boldsymbol{e}$, given that $\boldsymbol{y}$ was received. Provided the codewords are all equally likely, this strategy is optimum in the sense that it minimizes the probability of the decoder making a mistake and is called **maximum likelihood decoding**.

We now illustrate a table lookup technique for decoding systematic linear codes, which is a generalization of our example on the Hamming code. Define the $n \times (n-k)$ matrix H^T by

$$H^T = \begin{bmatrix} g_{1,k+1}\cdots g_{1n} \\ g_{2,k+1}\cdots g_{2n} \\ \vdots \\ g_{k,k+1}\cdots g_{kn} \\ 1\,0\,0\cdots\cdots 0 \\ 0\,1\,0\cdots\cdots 0 \\ \vdots \\ 0\,0\,0\cdots\cdots 1 \end{bmatrix} \tag{26.3}$$

The matrix H is called the *parity-check matrix*. Any code vector multiplied by H^T yields the null vector $\boldsymbol{x}H^T = \boldsymbol{0}$. Now consider postmultiplying any received vector $\boldsymbol{y}$ by H^T. The resulting $(n-k)$-dimensional binary vector is called the syndrome of the received vector and is given by $\boldsymbol{s} = \boldsymbol{y}H^T$. The decoding procedure is outlined as follows:

- Step 1. Initially, prior to decoding, for each of the 2^{n-k} possible syndromes $\boldsymbol{s}$, store the corresponding minimum weight vector $\boldsymbol{e}$ satisfying $\boldsymbol{e}H^T = \boldsymbol{s}$. A table of 2^{n-k} n-bit entries will result.

© 2006 by Taylor & Francis Group, LLC

- Step 2. From the n-dimensional received vector $\boldsymbol{y}$, generate one $(n-k)$-dimensional syndrome $\boldsymbol{s}$ by the linear operations $\boldsymbol{s} = \boldsymbol{y}H^T$; this requires an n-stage register and $n-k$ modulo-2 adders.
- Step 3. Do a table lookup in the table of step 1 to obtain $\hat{\boldsymbol{e}} = \boldsymbol{e}$ corresponding to the syndrome $\boldsymbol{s}$ found from step 2.
- Step 4. Obtain the most likely code vector by the operation $\boldsymbol{x}_m = \boldsymbol{y} \oplus \hat{\boldsymbol{e}}$. The first k symbols are the data symbols.

Examples of Linear Codes: Reed---Solomon Codes

After the discovery of Hamming codes, the development of the theory of **cyclic codes** provided the first large family of efficient error correcting codes (BCH codes). The central concept of cyclic codes is based on mathematical abstractions called *ideals* in the theory of finite fields. Cyclic codes are important practically because they can be implemented using high-speed shift-register-based encoders and decoders, at gigabit-per-second data rates. Within the family of cyclic codes there are specific classes of codes that are extremely powerful: Golay codes, BCH, and Reed–Solomon codes. Cyclic codes are defined in terms of a generator polynomial, which is multiplied by the data polynomial to obtain the codeword polynomial. These polynomials are defined in a finite field, and shift register circuits are used to perform the required polynomial multiplications and divisions. A popular family of codes called *cyclic redundancy check codes* (CRC) was derived from cyclic codes by *shortening* (reducing the block length) of some cyclic codes. CRC codes are primarily used for **error detection**, that is, to signal the presence of errors in the received message. Error detection, which is generally easier to obtain than error correction, is used in applications where it is extremely important to get the correct message or otherwise flag it as unreliable and discard it or ask for *retransmission*, according to a prescribed protocol as used in automatic-repeat-request (ARQ) systems. Reed–Muller codes (1954) are not used in practice any longer, but were the first codes to enable correction of multiple errors. They have fast decoding algorithms, but their rates are typically low. Reed–Muller codes are, however, very important ingredients for the construction of modern more complex codes.

Reed–Solomon (RS) codes, which can be viewed as a nonbinary extension of BCH codes, are a particularly interesting and useful class of linear block codes. The block length n of an RS code is $q-1$, with q being the alphabet size of the symbols. It can be seen that these codes are useful only for large alphabet sizes. RS codes with k information symbols and block length n have a minimum distance $d = n-k+1$, which is a property shared by all codes belonging to the class of *maximum distance separable* codes. RS codes are particularly effective on channels that produce errors in clusters, which are called *bursty* channels. Using symbols with $q = 2^m$ for some m, the block length is $n = 2^m - 1$. For an arbitrarily chosen odd minimum distance d, the number of information symbols is $k = n - d + 1$ and any combination of $t = (d-1)/2 = (n-k)/2$ errors can be corrected. If we represent each letter in a codeword by m binary digits, then we can obtain a binary code with km information bits and block length nm bits. Any noise sequence that alters at most t of these n binary m-tuples can be corrected, and thus the code can correct all bursts of length $m(t-1)+1$ or less, and many combinations of multiple shorter bursts.

26.6 Convolutional Codes

These codes can be generated by a linear shift-register circuit that performs a convolution operation on the information sequence. Convolutional codes were successfully decoded by **sequential decoding** algorithms in the late 1950s. A much simpler decoding algorithm—the **Viterbi algorithm**—was developed in 1967 and made low-complexity convolutional codes very popular. Unfortunately, this algorithm is impractical for stronger convolutional codes.

A rate $R = 1/n$ convolutional encoder is a linear finite-state machine with one input, n outputs, and a K-stage shift register, where $K = m+1$ is called the constraint length of the code and m is the memory

© 2006 by Taylor & Francis Group, LLC

of the finite-state machine. An encoder with $R = 1/2$ and $K = 3$ is shown in Fig. 26.4. Such an encoder has 2^m possible states and is described by the taps of the shift register that are used to form each output by modulo-2 additions. The set of taps of each output is described by a polynomial with binary coefficients, where a coefficient 1 indicates a tap and a coefficient 0 indicates no tap. The example in Fig. 26.4 shows how each input bit produces two output symbols, based on the present input and on two previous inputs that are stored in the shaded register cells after having been shifted in at previous ticks of the clock.

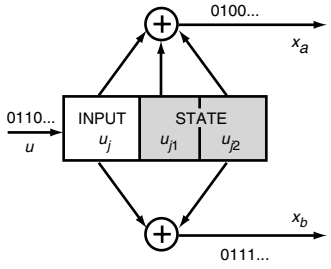


FIGURE 26.4 Example of convolutional encoder ($K = 3$).

A finite-state machine such as that shown in Fig. 26.4 can be described by a *state diagram*, as shown in Fig. 26.5(a). The state transitions are indicated by directed edges labeled by $u_j/x_{0j}x_{1j}$, that is, the input and the two outputs associated with a transition from a state $u_{j-1}u_{j-2}$ to state $u_j u_{j-1}$. The **trellis diagram** shown in Fig. 26.5(b) gives an equivalent description of the state transition of this finite-state machine stressing the evolution in time. Here a path traversing states from left to right corresponds to the evolution with time of the encoder. As for block codes, the properties of convolutional codes depend on the Hamming distances of possible output symbol stream, which correspond the codewords of the code. Good codes have large minimum distances, where the minimum distance can be determined by finding the minimum output weight (number of ones) of any path that starts in state 00 and end in the same state after some digression. The choice of a reference state is irrelevant due to the linearity of these codes. The trellis diagram shown in Fig. 26.5(b) was a crucial element in the discovery of an efficient method for decoding these codes: the Viterbi algorithm (1967), from the name of its inventor. This algorithm is an efficient method to compare the distance of a received symbol stream to that of all of the possible paths on the trellis and to select that which has the least Hamming distance. The efficiency of the algorithm is given by the fact that at each state (a node in the trellis) the accumulated distances of paths coming into the state can be compared, and only the path of least distance can be preserved. The other paths can be discarded forever without loss of optimality. This considerably reduces the amount of operations necessary to compute distances to all possible paths. Once this pruning of the trellis has been done, the surviving paths are the only candidates to be at minimum distance from the received sequence. If the trellis is terminated at some time, there will be only one survivor into state 00. Otherwise, as in common practice, if the sequence is semi-infinite, the survivor into the state with minimum accumulated distance will be selected. Once a surviving path is selected, the corresponding sequence of decoded bits can be read following the only possible backward path along the survivor. It has been shown (Forney, 1973) that the Viterbi algorithm implements, in fact, maximum-likelihood decoding, that is, the optimum decoding strategy.

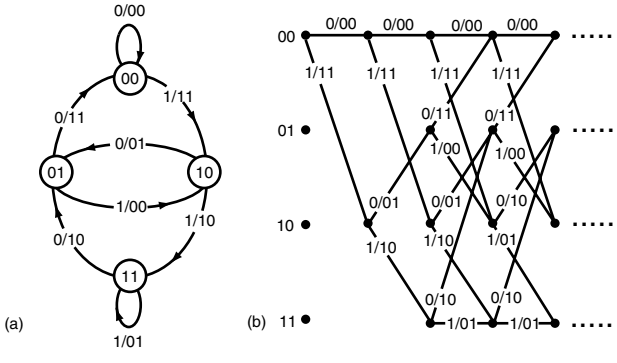


FIGURE 26.5 Examples of state and trellis diagrams.

© 2006 by Taylor & Francis Group, LLC

Sequential decoding is another method used earlier to decode convolutional codes: its complexity does not increase exponentially with the memory m of the code, but its performance can only approximate that of a true maximum-likelihood decoder, as the Viterbi decoder. Sequential decoding is a procedure for systematically searching through a *code tree* and using received information as a guide, with the objective of eventually tracing out the path representing the actually transmitted information sequence. The two best known sequential decoding algorithms are the *stack algorithm* and the *Fano algorithm*. Sequential decoding is an example of incomplete decoding which yields two types of decoding failure. The first, called an *undetected error*, occurs when the decoder accepts a number of wrong hypotheses and moves ahead anyway. The second, called a *buffer overflow*, occurs when the number of computations permitted per block is exceeded. In this case, the frame cannot be decoded, and is considered an erased or *deleted frame*. Convolutional codes, which are sequentially decoded, typically have a large enough constraint length so that the undetected error probability of the decoder is negligible compared to the probability that a block cannot be successfully decoded in the time allowed.

For applications requiring large coding gain and very low bit error probabilities the idea of cascading two codes has been widely used and called concatenated coding. The most powerful combination in use is that of an outer Reed–Solomon code with an inner convolutional code. The decoding is done in two stages, which makes it intrinsically suboptimal, but its performance is a good approximation of the optimum decoder. Since the inner decoder produces errors clustered in bursts it is usually necessary to use a technique called **interleaving** to scatter such errors and ease the job of the outer decoder. Typically, the inner Viterbi decoder corrects enough errors so that a high-rate outer code can reduce the error probability to the desired level.

Turbo Codes

Coding theorists have traditionally attacked the problem of constructing powerful codes by developing codes with a lot of structure, which lends to feasible decoders. However, coding theory suggests that codes chosen at random should be pretty good if their block size is large enough. The challenge to find practical decoders for almost random, large codes had not been seriously considered until recently.

In 1993 a new class of concatenated codes called turbo codes, was introduced. These codes, which can be viewed as convolutional or block codes, can achieve near-Shannon-limit error correction performance with reasonable decoding complexity.

A turbo encoder is a combination of two simple convolutional encoders. For a block of k information bits, each constituent code generates a set of parity bits. The turbo code consists of the information bits and both sets of parity, as shown in Fig. 26.6. The key innovation is an interleaver P, which permutes the original k information bits before encoding the second code. If the interleaver is well chosen, information blocks that correspond to error-prone codewords in one code will correspond to error-resistant codewords in the other code. The resulting code performs much like Shannon's random codes, which can approach optimum performance at the price of a prohibitively complex decoder. Turbo decoding uses two simple decoders matched to the constituent codes. Each decoder sends likelihood estimates of the decoded bits to the other decoder, and uses the corresponding estimates from the other decoder as *a priori* likelihoods. The turbo decoder iterates between the outputs of the two constituent decoders until reaching satisfactory convergence.

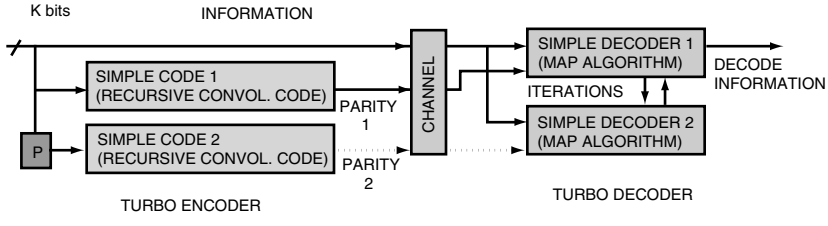


FIGURE 26.6 Example of turbo encoder/decoder.

© 2006 by Taylor & Francis Group, LLC

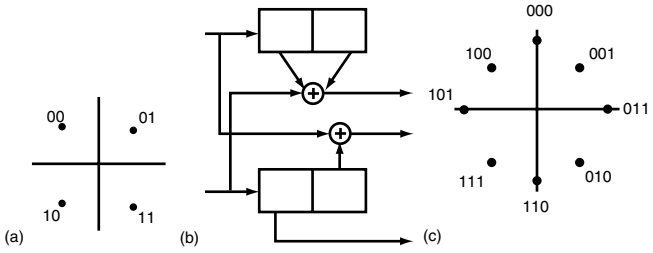


FIGURE 26.7 Example of trellis-coded modulation.

Turbo codes outperform even the most powerful codes known to date, but more importantly they are simpler to decode. To achieve their phenomenal performance, turbo codes require the use of large interleavers, but not much larger than those used by current concatenated codes.

26.7 Trellis Coded Modulation

Various efforts to improve the signal-to-noise ratio in channels with additive noise and *intersymbol interference*, which is characteristic of severely bandlimited channels (e.g., the telephone channel), led to the original development of *set partitioning* techniques that culminated in Ungerboeck's development (1982) of **trelliscoded modulation** (TCM).

Ungerboeck showed how large Hamming distance between differing data sequences does not necessarily imply large Euclidean distance between modulated data sequences, unless the assignment of coded signals to modulated signals is cleverly made. He proposed a new method of set partitioning, which not only provided a method for the assignment of coded signals to channel signals, but also provided a simple formula for the lowerbound of the Euclidean distance between modulated data sequences. Figure 26.7(b) and Figure 26.7(c) illustrate an example of encoder and modulated signal assignment for trellis-coded modulation, which we will compare to the popular uncoded quadrature phase shift keying (QPSK) scheme of Fig. 26.7(a).

The trellis code (Fig. 26.7(b)) replaces the QPSK signal constellation with an 8-ary PSK signal constellation so as to introduce eight points in the complex plane that represents phase modulation. Instead of using the eight points to increase the data rate by modulating three bits at a time into a channel symbol, the trellis code modulates only two bits of data at a time into a three-bit channel symbol. The three code bits define one of eight possibilities that are labeled in the phase diagram in Fig. 26.7(c). Every pair of incoming data bits is mapped into three code bits by the trellis encoder in order to create a waveform. The transmitted trellis-coded waveform has the same data rate as a QPSK waveform and uses the same bandwidth, but the power requirement is reduced by 2.5 times.

TCM can also be used to increase the transmitted data rate at a fixed transmitted power. Much more complicated signal constellations are required for this purpose.

26.8 Applications

Error correcting codes have been applied to a variety of communication systems. Digital data are commonly transmitted between computer terminals, between aircraft, and from spacecraft. Codes can be used to achieve reliable communication even when the received signal power is close to the thermal noise power. As the radio waves spectrum becomes ever more crowded, error-correction coding becomes an even more important subject because it allows communication links to function reliably in the presence of interference. This is particularly true in military applications, where it is often essential to employ an error correcting code to protect against intentional enemy interference.

© 2006 by Taylor & Francis Group, LLC

Error correcting codes are an excellent way to reduce power needs in communications systems where power is scarce, as in communication relay satellites because weak received messages can be recovered correctly with the aid of the code.

Data flow within computer systems usually is intolerant of even very low error rates, because a single error can destroy a computer program, and error correcting codes are becoming important in these applications. Bits can be packed more tightly into dense computer memories, by using simple Hamming codes. Codes are also widely used to protect data in digital tapes and disks.

Communication is also important within complex systems, where a large data flow may exist between subsystems, as in digital switching systems, and in digital radar signal-processing systems. These internal data might be transferred either by hardwired lines or by more sophisticated time-shared data bus systems. In either case, error correcting techniques are becoming important to ensure proper performance.

Defining Terms

Block codes: Codes that take an information block of length k and produce a codeword of length $n > k$ (n is the block length or code length).

Burst error correction: An error correction method that is effective against errors occurring in clusters, as opposed to random errors.

Channel capacity: The maximum information rate that can be transmitted across a channel, with arbitrarily low bit error rate.

Code rate: The ratio of the number of information bits k in a block to the total number of transmitted symbols n in a codeword ($n = k +$ no. of redundant symbols).

Codeword: The sequence of symbols obtained by adding redundant symbols to the original message.

Concatenated codes: Codes based on cascading two codes separated by an interleaver.

Convolutional codes: Codes generated by a linear shift-register circuit that performs a convolution operation on the information sequence.

Cyclic codes: Codes defined in terms of a generator polynomial, which is multiplied by the data polynomial to obtain the codeword polynomial. Polynomials are defined in a finite field, and shift register circuits are used to perform the required polynomial multiplications and divisions; examples: Golay codes, BCH, and Reed–Solomon codes.

Error detection: The mechanism by which a code can detect the presence of errors in a codeword. Error patterns that cannot be detected are called undetected errors.

Galois field: A Galois field (or finite field) is a set of objects or elements on which two operations $+$ and $\cdot$ are defined, obeying some specific rules (i.e., operations are commutative and distributive). Example: The integers $\{0, 1, \ldots, p-1\}$ modulo p form a finite field if p is prime.

Generator matrix: The $k \times n$ matrix that produces a codeword (of length n) of a block code, by multiplication by the information block (of length k).

Hamming distance: The number of places in which two sequences of equal length differ. The minimum distance of a code is the smallest Hamming distance between any pair of codewords.

Interleaving: A device that scrambles the symbols from several codewords so that symbols from a given codeword are well separated.

Linear codes: Codes such that the linear combination of any set of codewords is a codeword.

Maximum-likelihood decoding: A decoding procedure that maximizes the probability of a received sequence given any of the possible codewords. If codewords are equally likely, this decoding method yields the minimum possible error probability.

Redundancy: Additional symbols (parity check symbols) appended to the original message to allow for error correction.

Sequential decoding: A procedure for systematically searching through a code tree with the objective of approximating the solution for the path at minimum distance from the received sequence. Examples: Fano algorithm, stack algorithm.

Syndrome: A vector computed by the decoder to find the location of correctable errors.

© 2006 by Taylor & Francis Group, LLC

Systematic codes: Codes whose first k symbols of each codeword are identical to the information block. All codes can be reordered into systematic form.

Trellis coded modulation: A combined method for joint coding and modulation based on the design of convolutional codes matched to the modulation signal set in such a way to maximize the Euclidean distance between modulated sequence.

Trellis diagram: A graph used to represent the evolution in time of a finite-state machine by defining states as vertices and possible state transitions as edges.

Turbo codes: Codes generated by the parallel concatenation of two (or more) simple convolutional encoders, separated by interleavers.

Viterbi algorithm: An efficient method for decoding convolutional codes based on finding the path on a trellis diagram that is at minimum distance from the received sequence.

Weight distribution: The list of the Hamming distances of each codeword from a given reference codeword.

References

Berlekamp, E.R. 1968. *Algebraic Coding Theory*. McGraw-Hill, New York.

Berlekamp, R.R., ed. 1974. *Key Papers in the Development of Coding Theory*. IEEE Press, New York.

Blahut, R. 1983. *Theory and Practice of Error Control Codes*. Addison-Wesley, Reading, MA.

Bose, R.C. and Ray-Chaudhuri, D.K. 1960. On a class of error correcting binary group codes. *Info. and Control* 3:68–79.

Clark, G.C. and Cain, J.B. 1981. *Error-Correction Coding for Digital Communication*. Plenum, New York.

Fire, P. 1959. A class of Multiple-Error-Correcting Binary Codes for Non-independent Errors. Sylvania Report No. RSL-E-2, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, Calif., March.

Forney, G.D. 1973. The Viterbi algorithm. *Proc. IEEE* 61:268–278.

Forney, G.D. 1967. *Concatenated Codes*. MIT Press, Cambridge, MA.

Hamming, R.W. 1980. *Coding and Information Theory*. Prentice-Hall, Englewood Cliffs, NJ.

Hocquenghem, A. 1959. Codes correcteurs d'erreurs. *Chiffres* 2:147–156.

Lidl, R. and Niederreiter, H. 1983. *Finite Fields*. Addison-Wesley, Reading, MA.

MacWilliams, F.J. and Sloane, N.J. 1977. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam.

Muller, D.E. 1954. Application of Boolean algebra to switching circuit design and to error detection. *IEEE Trans. Computers* 3:6–12.

Reed, I.S. 1954. A class of multiple-error-correcting codes and the decoding scheme. *IEEE Trans. Info. Theory* 4:38–49.

Reed, I.S. and Solomon, G. 1960. Polynomial codes over certain finite fields. *J. SIAM* 8:300–304.

Schouhamer Immink, K.A. 1989. Coding techniques for the noisy magnetic recording channel—A state-of-the-art report. *IEEE Trans. Comm.* 37(5):413–419.

Schouhamer Immink, K.A. 1991. *Coding Techniques for Digital Recorders*. Prentice-Hall, Englewood Cliffs, NJ.

Shannon, C.E. 1948. A mathematical theory of communication. *Bell Sys. Tech. J.* 27:379:423, 623:656.

Ungerboeck, G. 1982. Channel coding with multilevel/phase signals. *IEEE Trans. Inf. Theory* (Jan.).

Viterbi, A.J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* IT-13: 260–269.

Viterbi, A.J. and Omura, J.K. 1979. *Principles of Digital Communication and Coding*. McGraw-Hill, New York.

Further Information

Most of the coding systems discussed in this chapter are too complex for a complete and detailed explanation. For in-depth treatment of general concepts in coding theory it is suggested to consult Clark and Cain (1981), Berlekamp (1974), Berlekamp (1968), MacWilliams and Sloane (1977), Blahut (1983), Viterbi and

© 2006 by Taylor & Francis Group, LLC

Omura (1979), and Hamming (1980). These references are a representative cross section of the literature on the subject of error correcting codes.

On the subject of coding for magnetic recording more details can be found in Schouhamer Immink (1989). The compact disk coding problem is discussed in Schouhamer Immink (1991). The origins of concatenated codes are treated in Forney (1967). The important mathematical tools of finite fields are treated in Lidl and Niederreiter (1983).

© 2006 by Taylor & Francis Group, LLC